Republic of Iraq

Ministry of Higher Education and Scientific Research

University of Babylon

College of Information Technology

Software Department

# *Twitter Bot Detection Using Improved Clustering Algorithms*

**A DISSERTATION**

**SUBMITTED TO THE COUNCIL OF THE COLLEGE OF INFORMATION TECHNOLOGY AT THE UNIVERSITY OF BABYLON IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTORATE OF PHILOSOPHY IN INFORMATION TECHNOLOGY/ SOFTWARE**

**By**

**RAAD GHAZI HAMEED KARIM**

**Supervised by**

**PROF. DR. SAFAA OBAYES MAHDI MADHLUM**

**2023 A.C.** **1445 A.H.**

!

﴿وَمِنَ ٱلنَّاسِ مَن يُعۡجِبُكَ قَوۡلُهُۥ فِي ٱلۡحَيَوٰةِ ٱلدُّنۡيَا وَيُشۡهِدُ ٱللَّهَ عَلَىٰ مَا فِي قَلۡبِهِۦ وَهُوَ أَلَدُّ ٱلۡخِصَامِ ٢٠٤ وَإِذَا تَوَلَّىٰ سَعَىٰ فِي ٱلۡأَرۡضِ لِيُفۡسِدَ فِيهَا وَيُهۡلِكَ ٱلۡحَرۡثَ وَٱلنَّسۡلَۗ وَٱللَّهُ لَا يُحِبُّ ٱلۡفَسَادَ ٢٠٥ وَإِذَا قِيلَ لَهُ ٱتَّقِ ٱللَّهَ أَخَذَتۡهُ ٱلۡعِزَّةُ بِٱلۡإِثۡمِۚ فَحَسۡبُهُۥ جَهَنَّمُۚ وَلَبِئۡسَ ٱلۡمِهَادُ ٢٠٦﴾

[ البقرة:204-206]

صدق الله العلي العظيم

## Supervisor Certification

I certify that this dissertation was prepared under my supervision at the Department of Software / Collage of Information Technology / Babylon University, by **RAAD**

**GHAZI HAMEED KARIM** as a partial fulfillment of the requirements for the degree of **Ph.D. in Information Technology**.

Signature:

Name: **PROF. DR. SAFAA OBAYES MAHDI**

Title: **Professor**

Date: / / 2023

## The Head of the Department Certification

In view of the available recommendation, we forward this dissertation for debate by the examining committee.

Signature:

Name: **DR. SURA ZAKI ALRASHID**

Title: **Assistance Professor**

Date: / / 2023

## Declaration

I hereby declare that this dissertation, (**Social Media Bot Detection** Using Improved Clustering Algorithms) submitted to the University of Babylon in partial fulfillment of requirements for the degree of Doctorate of Philosophy in Information Technology-Software has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for reports and summaries whose sources are appropriately cited in the references.

Signature:

Name: **RAAD GHAZI HAMEED KARIM**

Date: / / 2023

# Dedication

**To The Messenger of Mercy and Humanity**

**(Allah's Blessings be upon him and his household))**

**To his cousin, Imam Ali Ibn Abi Talib (AS)**

**To his daughter, Fatima Al Zahraa (AS)**

**To his grandsons, Imam Al Hassan and Imam Al Hussein (A.S)**

**Allah's Blessings be upon him and his household**

**To the savior of mankind …**

**Al-Imam Al-Mehdi (peace be upon him)**

**To My Father …**

**His soul, who left me alone for this year.**

**Who taught me that hard work does not take place except with patience and determination.**

**To my mother …**

**You have always been a source of kindness and compassion; I grew up on the shoulders of your patience.**

**To my wife and children**

**Who are the secret of my happiness in life.**

## Acknowledgments

To begin, I would like to offer my gratitude to Allah Almighty for granting me the ability to accomplish this task successfully. I also extend my utmost respect to the Commander of the Faithful and hero of Islam, born in the heart of the Kaaba,

Imam Ali Ibn Abi Talib, who is the cousin of the Prophet. Moreover, I express my gratitude to Imam Hussein, whose sacrifices gave me the strength to overcome challenging circumstances and complete this work.

I have been fortunate to receive invaluable support from many individuals during my Ph.D. journey, and I would like to acknowledge their contributions. I am deeply grateful to my supervisor, DR. SAFAA O. AL-MAMORY, for his guidance, supervision, and tireless efforts throughout the course of this work. I would also like to thank Dr. Ahmed Saleem Abbas for his full support during the courses and research phases, which helped me overcome numerous obstacles.

Furthermore, I wish to express my appreciation to all my colleagues at the College of Information Technology/Babylon University for their kind assistance during the research experiments, particularly Dr. Ahmed Habeeb Al-Azawi, Prof. Faez Ali AL-Maamori, Prof. Iman Saleh, Prof. Assad Sabah Hadi, Prof. Wisam Bahia, and Dr. Waddah Razouqi Abboud Hassan Buyi.

I cannot forget to acknowledge my family members, especially my mother, my sister, and my wonderful wife, for their unwavering support, patience, and love. Without them, I would not have been able to complete this process. I also thank my brothers for consistently supporting me in putting my research first. To my sons, who have been my constant source of inspiration, I extend my special appreciation for motivating me to complete this research quickly.

Finally, I would like to express my gratitude to all the kind, helpful, and lovely people who directly or indirectly helped me in completing this work. Although I am unable to mention them by name here, they are all in my heart, and I apologize for not being able to acknowledge them individually.

*Raad Ghazi Hameed Karim*

**Abstract**

Twitter faces the challenge of bot attacks, which can affect society. Acquiring labeled data for supervised machine-learning techniques that detect bots is expensive and time-consuming. Therefore, this dissertation proposes an unsupervised clustering technique to accurately detect bots on Twitter using unlabeled data. The aim is to provide a reliable and effective system for detecting bots. The proposed technique addresses the challenge of labeled data availability and seeks to achieve a balance between accuracy and computational complexity while handling outliers and optimizing memory usage.

The dissertation proposes an automated system that can detect Twitter bots which generally includes two key parts: the generation of new features and the bots prediction approach. Particularly, the proposed system consists of five phases: preprocessing, feature analysis, experiment design, techniques development, and system evaluation. The proposed features are developed using a combination of statistical and manual methods, and only the top-ranked ones are selected using the Correlation Attribute Eval technique. From the thirteen features generated, five features are important namely, NumberOfFollowings, CV_Following, pro, bfr-afr, and max-min. Moreover, three unsupervised machine learning algorithms which are KNN, DBSCAN, and stream K-means are modified to work automatically for bot detection by selecting input parameters for these algorithms based on unique data characteristics.

To assess the performance of the proposed features, three types of clustering algorithms: partition(k-means, k-medoid), hierarchy(agglomerative), and density(DBSCAN) are employed for bot detection. The results show that the proposed features are more effective than the original features for detecting bots. The success of this test of the proposed features announces the start of the second

part of our system which is the identification of bots using modified clustering algorithms.

The effectiveness of the proposed technique in detecting Twitter bots using clustering algorithms, particularly stream-based clustering algorithms, is confirmed in this dissertation. Multiple evaluation metrics are utilized in which high values are observed for the metric of Homogeneity (0.988), Completeness metric (0.989), V-measure metric (0.989), Adjusted Rand Index metric (0.996), Adjusted Mutual Information metric (0.989), Silhouette Coefficient metric (0.786), and Fowlkes Mallows Score metric (0.998).

This dissertation asserts that clustering techniques are cost-effective for labeling unlabeled Twitter data. Such techniques can offer a practical approach to identifying Twitter bots. The research highlights the significance of feature extraction and dimensionality reduction in improving clustering algorithms' performance. Integrating multiple unsupervised algorithms offers a more holistic approach to detecting Twitter bots. This research contributes to the field of bot detection and social media analytics, providing new possibilities for analyzing Twitter data.

DISSERTATION

# CONTENTS

DISSERTATION

# LIST OF ALGORITHMS

DISSERTATION

# LIST OF FIGURES

DISSERTATION

# LIST OF TABLES

DISSERTATION

# LIST OF ABBREVIATIONS

| Abbreviation | Meaning |
|---|---|
| SM | Social Media |
| SMB | Social Media Bots |
| ML | Machine Learning |
| UML | Unsupervised Machine Learning |
| DC | Dimensionality Curse |
| FE | Feature Extraction |
| SML | Supervised Machine Learning |
| FS | Feature Selection |
| NLP | Natural Language Processing |
| LDA | Latent Dirichlet Allocation |
| TF-IDF | Term Frequency-Inverse Document Frequency |
| IE | Information extraction |
| POS | Part-of-speech |
| GAWA | Genetic Algorithm and Wrapper Approaches |
| TM | topic models |

| | |
|---|---|
| **LSI** | latent semantic indexing |
| **TDM** | term-document matrix |
| **WEM** | word embedding |
| **PC** | Pearson's correlation |
| **IG** | information gain |
| **LA-MSBD** | LA-based malicious social bot detection |
| **CGAN** | conditional generative adversarial network |
| **GKDPCA** | the Gaussian kernel density peak clustering algorithm |
| **AUC** | Accuracy |
| **CIFA** | Community-Inspired Firefly Algorithm |
| **WSC** | Well-Separated Clustering |
| **ED** | Euclidean Distance |
| **MD** | Manhattan Distances |
| **MKD** | Minkowski Distance |
| **CBD** | Canberra Distance |
| **MinPts** | minimum points |
| **DC-LAKNN** | Locally adaptive k-nearest neighbor algorithm based on discrimination class |
| **AUC** | Area Under the Curve |
| **ST** | Standard Deviation |
| **UPF** | User Property Features |
| **UCF** | User Neighborhood Features |
| **UCF** | User Content Features |

| | |
|---|---|
| **CIFAS** | Community Inspired Firefly Algorithm with fuzzy cross-entropy |
| **DT** | Decision tree |
| **LR** | Logistic regression |
| **RF** | Random forest |
| **NB** | Naive Bayes |
| **SVM** | Support Vector Machines |
| **LA-MSBD** | Learning Automata-Based Malicious Social Bot Detection |
| **OCSVM** | One-Class Support Vector Machine |
| **PCA** | Principal Component Analysis |
| **LR** | Linear Regression |
| **BiLSTM** | Bidirectional Long Short-Term Memory |
| **$D^2$-sampling** | probability proportional to its squared distance to the nearest center |
| **DL** | Deep Learning |
| **CNN** | Convolutional Neural Networks |
| **CPB** | Contrast Pattern-Based Classification |
| **ISVM** | Improved Support Vector Machine |
| **kNN** | k-Nearest Neighbor |
| **DNN** | Deep Neural Network |
| **RNN** | Recurrent Neural Network |
| **DBSCAN** | Density-Based Spatial Clustering of Applications with Noise |
| **STM** | Stream k-means |

| | |
|---|---|
| **CAE** | Correlation Attribute Eval |
| **WSC** | Well-Separated Clustering |

# Chapter One

# General Introduction

## 1.1 Introduction

The pervasive use of Social Media (SM) has made it an indispensable aspect of our day-to-day lives, with millions of users around the world logging on to these platforms for communication, entertainment, and information sharing. SM such as Twitter, Facebook, and Instagram have revolutionized the way people communicate, interact, and consume content. The rise of SM has led to an increase in the power of user-generated content. SM platforms allow persons and establishments to share content with a wide audience, making it an essential tool for businesses, politicians, and celebrities to reach their target audience [1]–[3]. Furthermore, SM has become an important source of information, with news organizations using these platforms to share breaking news and updates. The term SM has been instrumental in shaping and guiding the general public's opinions, functioning as a critical platform for social movements and activism. However, with the increasing number of users and the growing impact of SM, the use of Social Media Bots (SMBs) has also risen, leading to concerns over their impact on user experience and engagement [3].

SMBs are programs of computers that generate content and interact with users, to influence users' behavior or spread certain messages [3]. Research has shown that Bots contribute significantly to a substantial portion of online activity, with Twitter disclosing that approximately 8.5% of its users are bots [4]. Studies have also revealed that a significant number of Twitter users who communicate in English and remain actively engaged display characteristics resembling those of bots. [5].

While some bots have benign intentions, others can be malicious, engaging in activities such as spamming or spreading false information. Therefore, detecting and identifying bots has become a pressing concern for both SM platforms and users. The ability to distinguish between bots and human users can help maintain the

integrity of the SM ecosystem and improve user experience. Bots can be detected using either classification or clustering techniques.

Clustering is a popular technique in data analysis that aims to identify useful groups of objects in a dataset based on their similarities. These groups are known as clusters, and their definition can vary depending on the goals of the analysis. There are numerous different concepts of a cluster that can be beneficial in practice. The four types of clusters are well-separated, prototype-based, graph-based, and density-based [6].

This dissertation aims to propose a new method to detect bots on SM platforms. It also examines the challenges associated with detecting bots and proposes a novel solution to improve the accuracy of bot detection. Specifically, this research focuses on developing new features and a hybrid clustering algorithm that does not require labeled data to detect bots.

### 1.2 Categories of Malicious Bots

SMBs can be classified into various categories based on their functionalities and purposes [7]. Understanding these different types of malicious SMBs is crucial in developing effective strategies to detect and mitigate their impact on SM platforms. Here, a brief explanation for each type of malicious SMB is provided to facilitate a better understanding of their characteristics and potential risks.

1) Cashtag piggybacking bots: These bots exploit popular hashtags and cashtags (symbols used to identify a stock or other financial security) to promote fraudulent or illegal activities [8].
2) Spam bots: These bots flood SM platforms with unwanted and often fraudulent content, such as scams, phishing attempts, and clickbait [9].

3) Astroturfing bots: These bots create simulate or fabricate the impression of grassroots support for a particular cause or opinion, but in reality, they are automated accounts created to manipulate public opinion [10].
4) Sybils: These bots use multiple fake accounts to create the impression of a larger group of users, which can be used to amplify a message or support a particular agenda [11].
5) Fake accounts used for botnet command and control: These bots are used to control a network of infected computers (a botnet) through SM accounts created for this purpose [12].
6) Pay bots: These bots are paid by individuals or organizations to artificially inflate the popularity of their content, such as by liking or retweeting their posts [13].
7) Social botnets in political conflicts: These bots are used to spread propaganda, manipulate public opinion, and create discord in political conflicts [14].
8) Infiltration of an organization: These bots are used to infiltrate organizations or businesses to steal sensitive information or disrupt operations [15].
9) Influence bots: These bots are used to influence the opinions or actions of other SM users by engaging with them, retweeting their posts, or posting content that supports a particular agenda [13].
10) Doppelganger bots: These bots create fake accounts that impersonate real users or organizations, often spreading false information or sowing discord [16].

The aforementioned classifications of malevolent bots are not very separate. Certain categories of malicious bots, such as botnets and Influence bots in contentious political scenarios may share similar goals and strategies, which complicates their classification based on unique features. To address this challenge, Table 1.1 provides a comprehensive overview of the primary SMBs classes, accompanied by a summary explanation of each category and the corresponding types that align with these observations. In other words, the classification of malicious bots is based on the classes that share similar goals and strategies, rather than solely relying on their unique features.

**Table 1.1: The Categories of Malevolent Bots Found on SM Platforms**

| Class | Types | Brief description | Statistics Usage of Bots | Ref. |
|---|---|---|---|---|
| Spam bots | 1, 2 and 6 | These bots disseminate harmful links, and unwanted messages, and take over popular SM topics [9]. | [17]–[20] According to these sources, the percentage of Twitter accounts that are spam bots ranges from under 5% to 15%. | [7], [21]–[24] |
| Social bots | 3, 7, 8 and 9 | These are software applications that are intended to utilize SM platforms by imitating human communication and interaction [14]. | According to a specific citation [25], Algorithms detecting bot activity suggest a proportion of around 15%. Another reference [20] indicates that estimates provided by researchers suggest that automated accounts, or bots, make up around 9% to 15% of the vast number of Twitter profiles. These estimations are based on an early study conducted in 2017 and more recent investigations conducted by a firm specializing in monitoring online discussions. | [7], [26]–[28] |
| Sybils | 4, 5 and 10 | These are fake or anonymous identities, such as user accounts, that exert an outsized influence on a given platform [11]. | According to references [20] and [29], the Israeli technology company Cyabra has provided an estimation of 13.7% for the proportion of inauthentic Twitter profiles, encompassing Sybil bots. | [7], [22], [29] |

| | | | | |
|---|---|---|---|---|
| Cyborgs | Any of the above types | These accounts can either be operated by humans utilizing automated methods or managed by human operators controlling bot accounts [30]. | The search results do not provide any specific statistics on the usage of Cyborg bots. Nonetheless, as cited in reference [20], the Israeli technology company Cyabra has provided an appreciation of 13.7% for the proportion of deceptive Twitter profiles, encompassing Cyborg bots. | [31]–[33] |

Utilizing this information can aid in the creation of a dynamic and adaptive system for bot detection, which can efficiently handle outliers and optimize memory usage while balancing accuracy and computational complexity. Such a system can prevent the spread of misinformation and improve online conversations, maintaining the integrity of SM. By leveraging this information to develop effective detection algorithms, the propagation of false information can be prevented and the quality of online interactions can be enhanced and this, in turn, can lead to preserving the credibility of SM.

### 1.3 Problem Statement

The evolution of bots represents a challenge in the field of Machine Learning (ML) [34]. As technology advances, botmasters are continually developing more sophisticated bots with changing features to evade detection. This presents a vital problem in social media. Nevertheless, the scarcity of extensive, precisely labeled datasets further intensifies this challenge.

One of the significant trade-offs in developing algorithms to detect bots is the balance between the number of features, accuracy, computational complexity, and speed. The more features used, the higher the accuracy of the algorithm, but it also results in increased computational complexity, which can impact the speed of the

algorithm. Using fewer features, however, can lead to faster algorithms, but with lower accuracy.

Unsupervised Machine Learning (UML) approaches such as clustering algorithms are effective in detecting bots. However, the key problem in such algorithms is the absence of assumptions about the number of clusters, the ability to handle outliers, the complexity of computational time, and the optimum use of memory. These issues make the development of efficient and accurate clustering algorithms for bot detection an open problem.

## 1.4 Research Aims and Objectives

The main aim of this dissertation is to develop an unsupervised clustering technique that can accurately detect bots on SM networks, specifically on Twitter. This aim can be covered by developing an efficient and accurate clustering algorithm for bot detection that can handle the evolving features of bots while utilizing unlabeled data to overcome the lack of accurately annotated datasets. The algorithm should strike a balance between the number of features, accuracy, computational complexity, and speed while effectively handling outliers and optimizing memory usage. To achieve this aim, several sub-objectives need to be covered, which include the following:

1) Comparing human attributes with those of Twitter bots to determine the most useful characteristics for bot detection.
2) Identifying the best features and extracting new features that can improve the performance of the detection system and reduce the number of dimensions.
3) Selecting the top-ranked features to enhance the performance of clustering algorithms and reduce dimensions.

4) Developing an integrated clustering algorithm that combines different clustering methods to ameliorate the accuracy of Twitter bot detection.

Generally, the dissertation strives to develop a reliable and efficient system for detecting bots on Twitter platforms. Such a system can contribute to curbing the dissemination of misinformation and enhancing the overall quality of online discussions.

## 1.5 Contributions of Dissertation

This dissertation adds several contributions to the area of bot detection on SM sites, which overcome the challenge of obtaining labeled data by relying on UML algorithms. The developed system uses three algorithms to independently select input parameters and provide an adaptive bot detection. New features have been developed to provide better insights into the characteristics of malicious bots, and their reliability has been tested by applying them to three clustering algorithms with promising results. This achievement provides confidence in the proposed system's effectiveness in accurately detecting bots on Twitter. The dissertation makes the following notable contributions:

1) Three unsupervised algorithms namely, KNN, DBSCAN, and stream K-mean are integrated to identify Twitter bots. As such, a new integrated approach is developed. The rationale behind this integration is based on the strengths and weaknesses of each algorithm to be overcome by the other technique.
   A. KNN was adapted by addressing its limitation (determine the number of neighbors(k)). k was calculated based on a developed equation.
   B. The limitation of DBSCAN was addressed by computing the optimal minimum samples (MinPts) and epsilon(ε) parameters automatically.
   C. Using the output of the DBSAN algorithm, the number of clusters was determined, and the distance approach was chosen. After that, a stream k-

mean algorithm was proposed to minimize computational time complexity and make the best use of memory.

2) The proposed system addresses a major challenge in the analysis of Twitter data, which is the difficulty in obtaining labeled data. The labeling process requires significant effort, cost, and time, as it involves distinguishing between human users and bots. To overcome this challenge, the system relies on UML algorithms, which are capable of improving over time. This is in contrast to supervised ML algorithms, which require pre-classified data to train models.

3) This dissertation proposes a groundbreaking system for the automatic detection of Twitter bots to enhance the role of bot detection, reduce dependence on users, and minimize errors. The system utilizes three unsupervised algorithms that independently select input parameters based on the data's unique characteristics. By combining these algorithms, a dynamic and adaptive approach to bot detection is achieved, resulting in unparalleled accuracy and efficiency.

4) To increase the accuracy of the unsupervised algorithms, new features were developed that provide better insights into the characteristics of malicious Twitter bots. These features are more reliable than previous ones and have been specifically designed to capture key indicators of bot activity, such as the frequency and timing of posts, the use of certain keywords and hashtags, and patterns of engagement with other users. By incorporating these features into our analysis, we can more accurately distinguish between genuine human users and malicious bots, ultimately improving the overall effectiveness of our bot detection system.

5) To ensure the reliability of the newly generated features and to avoid potential issues with bot detection, they were applied to three well-known types of

clustering algorithms. The results were highly promising, with a precision of 0.99, indicating that the features were effective at accurately distinguishing between genuine human users and malicious bots. This achievement gives us confidence in the reliability and effectiveness of our proposed system, which incorporates these features to achieve unparalleled accuracy in bot detection on Twitter.

### 1.6 Bot Detection Challenges

With the ongoing expansion in the usage of SM, so does the prevalence of bots, which are automated accounts designed to mimic human behavior on these platforms. However, detecting these bots can be a complex and challenging task, as they can exhibit complex behaviors, constantly evolve their tactics to avoid detection, and vary widely in their types and purposes. Additionally, limited access to data, international and cultural variations, scalability issues, and a lack of clear definitions for what constitutes a bot all contribute to the difficulty in detecting these accounts. The following explores these challenges in more detail:

1) Human-like behavior: Bots can exhibit complex behavior and mimic human-like interactions, making it challenging to distinguish them from real users [35]. Bot creators are constantly updating and changing their tactics to avoid detection, making it difficult for detection algorithms to keep up [35].
2) Variability in Bot Types: There are many different types of bots, ranging from simple to highly sophisticated, each requiring a different detection approach [7].
3) Scalability: SM platforms generate a massive amount of data, making it challenging to develop detection algorithms that can scale effectively.
4) Lack of clear definition: There is no clear definition of what constitutes a bot, making it challenging to distinguish between human and bot activity [36], and

most public datasets used for SMB detection only include some types of SMB. [35].

5) Intent identification: Identifying the intent behind bot activity can be difficult, as bots can be used for both benign and malicious purposes.
6) Feature engineering: The challenge in detecting SM bots lies in feature engineering, which involves selecting and extracting relevant features from data to build a machine-learning model that can differentiate between a bot and human behavior on SM platforms [36].
7) False positives: Overly aggressive bot detection can result in false positives, identifying real users as bots [37].

## 1.7 Explanation of Limitations

Detecting Twitter malicious bots is a challenging task due to several limitations that are expected to be faced in the dissertation. One of the significant limitations is that the proposed system will only focus on detecting bots on Twitter while ignoring bots in other SM networks or web-based systems. This could limit the effectiveness and scope of the system, as bots can operate across multiple platforms, and a bot detection system designed for Twitter may not apply to other platforms.

Another limitation is the availability and quality of the dataset used for training and testing the proposed system. Obtaining or building a sufficient dataset of high-quality data is a challenge in any ML project, and this project is no exception. The dataset must be large enough to represent the diversity of Twitter bot behavior and provide enough examples for the algorithm to learn from. Additionally, the dataset must be labeled correctly with the appropriate ground truth labels to evaluate the performance of the system accurately.

Moreover, in the case of UML techniques, applying them directly to a classification or regression issue is not feasible due to the absence of known output values. Consequently, training the algorithm in the conventional manner used in supervised learning becomes impossible. Unsupervised learning methods are used to discover patterns or groupings within data without any prior knowledge of the output values. Therefore, selecting the appropriate unsupervised learning method to identify patterns and classify Twitter malicious bots is a challenge that needs to be addressed.

## 1.8 Statement of Dissertation's Significance

The detection of malicious bots on SM platforms, particularly Twitter, is of significant importance for several reasons. A diverse range of objectives can be accomplished through the use of these bots, such as disseminating false information, intensifying specific messages, and manipulating the viewpoints of the general public. As such, their detection is crucial for ensuring the integrity of SM and preventing the spread of harmful or misleading information.

One of the key challenges in detecting malicious bots on Twitter is the sheer volume of data generated on the platform. With millions of users and tweets being posted every day, it can be difficult to identify patterns of bot activity that may be indicative of malicious intent. Additionally, many bots are designed to mimic human behavior, making it difficult to distinguish them from genuine users.

Despite these challenges, there have been significant advances in the development of ML algorithms and other detection techniques that can be used to identify malicious bot activity on Twitter. These techniques typically involve analyzing various features of Twitter data, such as the content of tweets, the timing and frequency of posts, and the relationships between users, to identify patterns that may be indicative of bot activity.

The significance of detecting malicious bots on Twitter extends beyond just the SM platform itself. The spread of misinformation and manipulation on Twitter can have real-world consequences, such as influencing public opinion and even impacting political outcomes. By detecting and removing these bots, it is possible to mitigate the potential harm that they can cause and ensure that Twitter remains a platform for genuine human interaction and communication. Generally, the detection of malicious bots on Twitter is a complex and ongoing challenge, but one that is of significant importance for maintaining the integrity of SM and preventing the spread of harmful or misleading information.

## 1.9 Literature Review

Lee et al. in [38] and Wang [9] developed an approach for identifying Twitter bots using various features related to the content, tweet account, and account usage. Lee et al.'s method, called Decorate, reached an F1 score of 0.88, while Wang achieved an F1 score of 0.91.

The Community Inspired Firefly Algorithm for Spam Detection (CIFAS) in [39] used fuzzy cross-entropy to maintain the same level of information in the selected feature subset as in the entire feature set, achieving an accuracy of 95.3%.

Botometer, suggested by Yang et al. [40] as it employs a Random Forest classifier trained on 1200 features and achieved an AUC score of up to 1.0 when classifying political bots, demonstrating a cross-validation performance of 0.97 AUC across all types of bots.

Shukla [41] suggests a Twitter bot detection framework that employs profile metadata. The study recommends using Weight of Evidence encoding, an extra-tree classifier for feature selection, and blending with the RF algorithm to achieve a 93% AUC.

Two clusters were identified after examining data from the Canadian Elections held in 2019 on Twitter [42]. The first one consisted of human users, whereas the other included bots. Unsupervised techniques were used, involving PCA and K-means, to create these clusters based on three features. The results suggest that 94% of the total Twitter activity was generated by human users, while the remaining 6% was generated by bots. Traditional detection of bot methods-based ML often suffers from imbalanced datasets and classifier prejudice, leading to poor detection rates of samples representing small.

In [43], the authors proposed an improved approach using a CGAN to generate additional data and address the imbalance issue. They also introduced a modified clustering algorithm, the GKDPCA, to generate an auxiliary condition for the CGAN and improve convergence. Experimental results show that their method outperforms traditional oversampling techniques and achieves a high F1 score of 97.56%, indicating its effectiveness in the field of social bot generation.

Chavoshi et al. [44] created an unsupervised approach called DeBot that employs cross-user behavioral correlations to identify Twitter bot accounts. The DeBot has the capability of detecting a large number of bot accounts daily with a precision rate of 94%, and it produces daily online reports.

The study [45] successfully identified harmful retweeting bots on Twitter by introducing a bot detection method called Retweet-Buster (RTbust). By employing unsupervised Feature Extraction (FE) and clustering techniques, the proposed approach was able to detect suspicious retweeting patterns and label them as bots. This allowed us to identify malicious retweeting activity on the platform. RTbust achieved remarkable results, with an F1 score of 0.87.

Based on the above discussion, there is still potential in exploring the topic of the public's attitudes toward AI in advertising as highlighted in earlier literature [46]. LDA-based topic modeling was utilized to analyze Twitter conversations about AI in advertising from 2018 to 2019 [46]. It identified eight distinct clusters of tweets on the subject, with the most favorable being related to "Marketing tools that are powered by AI technology," while the most unfavorable centered around " The involvement of AI in social media advertising efforts."

Shi, Zhang, and Choo [47] proposed the use of click-stream sequences as a reliable feature that is difficult for social bots to imitate and applied semi-supervised clustering to identify malicious bots. They found that click-stream sequences are effective in capturing shifts in a user's behavior and concealing their primary characteristics.

Dorri et al. [48] introduced SocialBotHunter, which utilizes the homophily property of social network graphs to identify spam bots on Twitter. The model considers users' social behavior and interactions to detect bots and can operate on a dataset that includes only labeled legitimate users as a seed.

Table 1.2 displays a summary of the comparative analysis conducted in the previous machine-learning literature. As shown, the majority of methods are used to detect Twitter bots that currently rely on supervised classifiers, particularly the Random Forest method, which has achieved a perfect AUC=1 score for certain bot types [40]. Existing Twitter bot detection methods require labeled examples of bot accounts to identify characteristic behavior. However, the diversity of bot behavior in the bot landscape makes it challenging to identify certain types of bots, particularly those that imitate human social interactions [21]. As bots continuously evolve their strategies to avoid detection, it is essential to develop Twitter bot

detection methods that do not solely rely on existing bots, as the behavioral patterns observed in these accounts may become obsolete over time.

**Table 1.2: Previous ML Works Comparison**

| Ref. | Performance | Method | Relevant Details | Approach |
|---|---|---|---|---|
| [38] | F1 = 0.87 | Decorate classifier | Tweet text account, and account usage features. | Supervised |
| [39] | AUC =90.88 | SVM, KNN, RF | An adapted version of the Firefly algorithm. | Supervised |
| [40] | AUC = 1 | Botometer (Random Forest-based) | Utilized 1200 features from various categories and trained with different bots. | Supervised |
| [41] | AUC =0 .93 | RF | The study concludes that Weight of Evidence encoding, and blending with RF are effective. | Supervised |
| [42] | | K-means Algorithm | Suggested that 94% of the total Twitter activity was generated by human users, while the remaining 6% was generated by bots. | Unsupervised |
| [43] | F1 =0. 97 | GKDPCA | Uses CGAN to generate additional data and address the imbalance data issue. | Unsupervised |
| [44] | Precision=0.94 | DeBot: Works with principle(Stream of Activities, and Hashing Technique) | Utilizes correlations in Interactions between multiple users to detect bot accounts on Twitter. | Unsupervised |
| [45] | F 1 = 0.87 | Hierarchical Density-Based Algorithm | The study effectively detected harmful retweeting bots using RTbust and unsupervised feature engineering and clustering. | Unsupervised |
| [46] | Coherence score= 0.52 | LDA-based | This research utilized NLP within Python. | Unsupervised |
| [47] | Recall= 0.98 | K-means algorithm | Applying semi-supervised clustering to identify malicious bots | Semi-Supervised |
| [48] | Recall= 99.08 | SocialBotHunter | The unified model integrates both the social graph's structural information and the users' social behavior information. | Semi-Supervised |

## 1.10 Dissertation Outline

This dissertation is composed of five chapters, which are outlined in this section (refer to Figure 1.3 below for a visual representation).

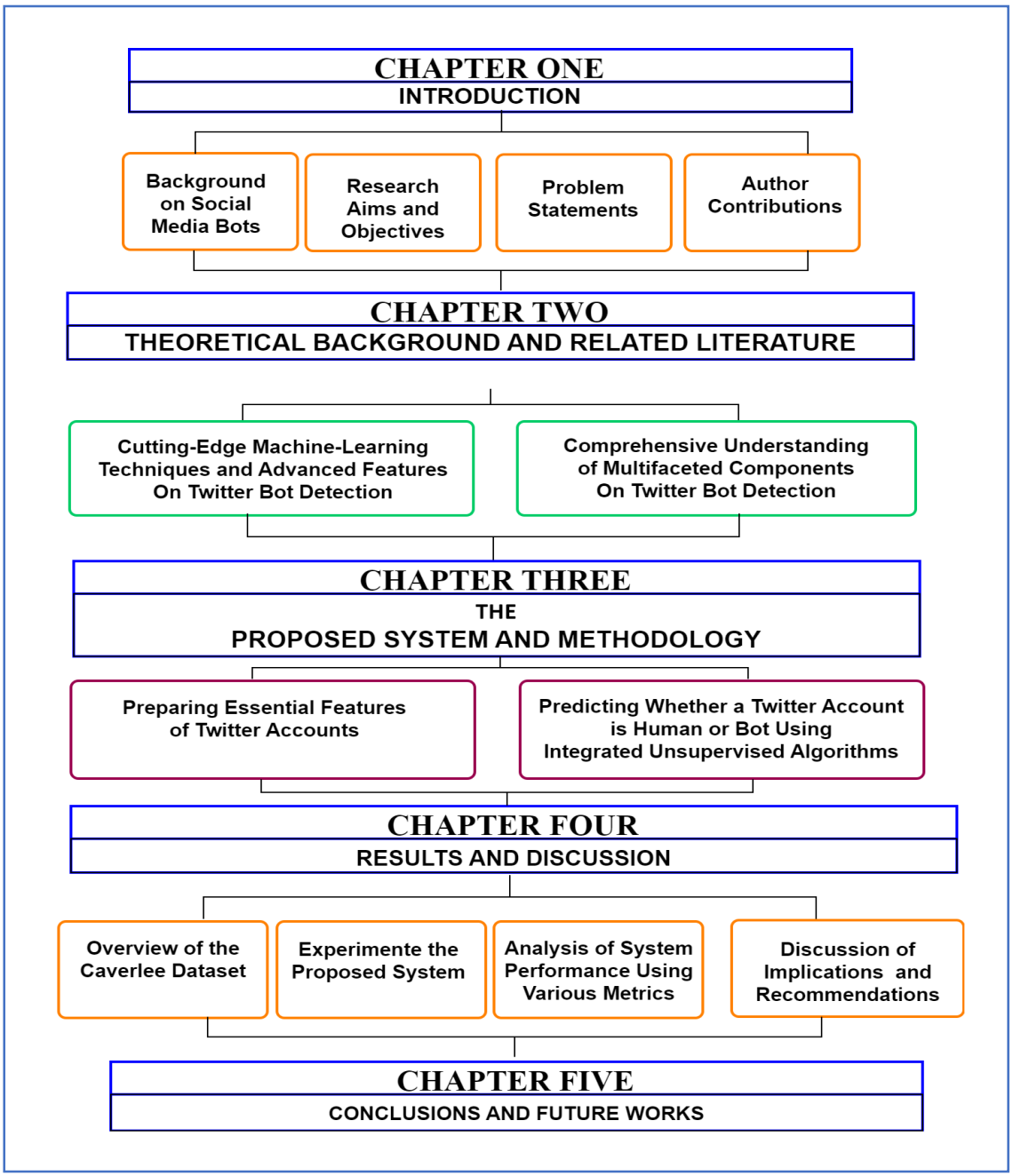



**Figure 1.1: The Dissertation Flow**

**Chapter Two:** provides an overview of the current state-of-the-art clustering algorithms for Twitter bot detection and explores recent research on feature engineering and selection. By highlighting the significance of feature selection and engineering, this survey aims to develop more effective Twitter bot detection systems to identify and prevent malicious behavior on the platform.

**Chapter Three:** clarifies the main steps of designing an integrated Unsupervised clustering algorithms system for the Twitter bot detection problem and with their top rank features.

**Chapter Four:** shows the employment of the proposed system on the Carvelli dataset after the labels were removed from it and the experimental results of this implementation.

**Chapter Five:** lists the derived conclusions of this dissertation and provides suggestions for future work.

## 1.11 Summary

The focus of this dissertation is on developing an unsupervised clustering technique that accurately detects bots on Twitter. The goal is to provide a reliable and effective system for detecting bots on SM networks, thereby preventing the spread of misinformation and improving online conversations. The problem statement of the dissertation is to develop an efficient and accurate clustering algorithm for detecting evolving bots, given the challenges of limited datasets, computational complexity, and speed. The algorithm must balance accuracy and computational complexity while effectively handling outliers and optimizing memory usage. Developing such an algorithm would be a significant contribution to the field of ML.

The dissertation proposes a system for the automatic detection of Twitter bots that overcomes the challenge of obtaining labeled data by utilizing UML algorithms. The proposed system uses three algorithms to achieve dynamic and adaptive bot detection, and new features are developed to enhance the accuracy of the unsupervised algorithms. The proposed system provides significant contributions, including addressing the challenge of obtaining labeled data, proposing a dynamic and adaptive approach to bot detection, developing novel features for better insights into malicious bot activity, and achieving unparalleled accuracy in bot detection on Twitter.

However, the system's focus is limited to Twitter bots and not on bots operating on other SM platforms or web-based systems. Additionally, the availability and quality of the dataset used for training and testing the system pose a challenge. Selecting an appropriate UML method is also a challenge. The text emphasizes the need for careful planning, execution, and continuous monitoring to overcome these limitations and ensure the system's success.

In conclusion, the proposed system provides a solution to the challenge of detecting malicious bots on Twitter. By detecting these bots, the system can mitigate the potential harm they can cause and ensure that Twitter remains a platform for genuine human interaction and communication, and help classify real Twitter data, which takes a lot of time and cost to label. The suggested system makes a substantial contribution to the area of ML and has the capability of being expanded to other SM platforms or internet-based systems.

# Chapter Two

# Theoretical Background and Related Literature

## 2.1 Overview

Twitter has become a crucial information dissemination and communication platform, with over 300 million monthly active users[49], [50]. However, this popularity has also attracted the attention of malicious users who exploit the platform to spread misinformation, spam, and engage in cyberbullying. One of the ways to mitigate these negative effects is through the use of Twitter bot detection techniques.

This chapter delves into the cutting-edge machine-learning techniques and advanced features employed in the realm of Twitter bot detection. Its main goal is to provide a comprehensive perception of the multifaceted components inherent in Twitter bot detection, enabling the development of robust systems that effectively mitigate malicious activities.

Table 2.1 provides a structured overview of how this Chapter presents the information regarding Twitter bot detection based on previous literature reviews. It serves as a valuable roadmap, organizing the key points and components discussed throughout the chapter. It focuses on cutting-edge machine-learning techniques and advanced features in bot detection. Afterward, it dissects the main elements covered, which consist of Machine Learning (ML) algorithms and approaches, selecting and engineering features for precision, and the significance of certain Twitter features in detecting bots. This Chapter also addresses the significance of ML in augmenting the identification and characterization of Twitter bots. Furthermore, it outlines the different ML types utilized, such as supervised and unsupervised learning, and their respective roles in bot identification and pattern discovery. Summarizing these crucial aspects in a structured format allows readers to navigate and comprehend the key concepts of this area effectively, providing a clear and organized presentation of the information gathered from previous literature reviews.

**Table 2.1: Overview of Twitter Bot Detection Based on Literature Reviews**

| Section | Key Points |
|---|---|
| **Chapter title** | Concepts and Literature Reviews for ML Techniques and advanced features in Twitter Bot Detection. |
| **Objective** | Provide a comprehensive understanding of the multifaceted components of Twitter bot detection |
| **Key Components** | 1. Cutting-edge ML algorithms and approaches and methodologies<br>2. Feature engineering and selection for accuracy<br>3. Important Twitter features for bot detection |
| **Significance of ML** | 1. Augments identification and characterization of Twitter bots<br>2. Enhances accuracy and efficacy of bot detection systems |
| **ML Types** | 1. Supervised learning algorithms for bot identification<br>2. Unsupervised learning methods (clustering algorithms and anomaly detection) for discovering emerging bot patterns and anomalous behaviors |
| **Important Twitter Features** | 1. Account-based features:<br>- Account age<br>- Account verification status<br>- Number of followers and followings<br>- User description<br>- Profile image characteristics<br>2. Tweet-based features:<br>- Posting frequency and time patterns<br>- Retweet and favorite counts<br>- Mention and hashtag usage<br>- URL and media attachments<br>- Sentiment and emotion analysis<br>- Language and syntax patterns<br>3. Network-based features:<br>- User interaction network<br>- Network centrality measures<br>- Community detection |
| **Model Selection** | Involves the selection of appropriate models for Twitter bot detection |
| **Comprehensive Detection Frameworks** | Equips practitioners with the knowledge to develop comprehensive and robust detection frameworks |

## 2.2 Bots Detection Technologies

Twitter is a popular social media platform that has been plagued by the presence of bots, which are often used to spread misinformation, spam, and other types of malicious content. To address this problem, researchers and developers have proposed various detection technologies based on different approaches graph-based,

Anomaly-based, machine-learning approaches, and crowdsourcing-based approaches [51]–[53]. The graph-based approach analyzes the structure of social networks to detect abnormal patterns that may indicate the presence of bots.

The crowdsourcing approach relies on human annotators to identify bots based on specific criteria. The feature-based approach uses ML algorithms to analyze highly relevant features that distinguish between bots and humans, such as linguistic patterns and timing of interactions. Finally, the anomaly-based approach is premised on the idea that typical users of social media networks are unlikely to engage in uncommon behaviors that offer no benefits. ML algorithms are called "feature-based" because they rely on the selection and extraction of relevant features from the input data. These features are essentially the input variables that the algorithm uses to learn and make predictions. In other words, the algorithm tries to learn a mapping between the features and the output variable. Feature engineering is the procedure of choosing and extracting the most pertinent features from the input data. This process can greatly impact the performance of the ML algorithm. A well-engineered set of features can lead to better accuracy, while a poorly chosen set of features can result in poor performance [54].

The suggested approach in this dissertation takes a variety of ML advantages to detect Twitter bots, which provides several benefits such as speed, accuracy, scalability, and adaptability [55]. Moreover, the ML algorithms can be trained using both labeled (supervised) and unlabeled (unsupervised) data, enabling them to identify bots without human intervention.

### 2.2.1 Crowdsourcing-Based Bot Detection

Crowdsourcing is an online task where numerous individuals are invited to perform a task manually [56]. Although this technique can be effective for detecting

small and medium-sized businesses, it can also be costly in terms of either time or financial resources, some researchers have employed it to obtain labeled datasets that can be utilized for further research. [56]. This approach entails gathering work, information, or feedback from a significant number of individuals who submit their responses through the Internet, social media, and smartphone applications[57].

In [58], a significant contribution was added to this field by conducting a social Turing test to discern whether an Online Social Networks (OSN) account is a Sybil. The proposed system employs automated algorithms to filter through accounts and identify those that seem suspicious, which are subsequently examined by a carefully selected group of crowdworkers. The collective inputs of these crowdworkers are consolidated to enhance the accuracy of Sybil detection. Moreover, a ground truth dataset was created by manually categorizing Twitter accounts as either humans, Sybils, or Cyborgs [59]. Ten skilled volunteers were recruited to work as crowdworkers for this challenging task.

Another research study [60] developed a new approach called Bot-Detective to identify bots on Twitter. This approach is responsible and interpretable, using an explainable ML framework and a crowdsourcing module to gather feedback from users. A publicly available annotated dataset was created using Twitter's rules and existing tools, and in situ, experimentation showed that Bot-Detective is accurate and has the potential to be scaled up as a service. However, we believe that the method could be improved by increasing training data volume, exploring different feature selections (FS) for different types of bots, and evaluating and comparing other explainable AI methods. It was also suggested that improve the explanations provided by Bot-Detective be more specific or generalized depending on the situation. Despite the tool's potential, further research and enhancements are necessary to overcome limitations and increase efficiency.

### 2.2.2 Graph-Based Bot Detection

Structures of graphs are frequently utilized to depict social network formations, where nodes depict individuals or entities, and edges depict the connections or ties between them. Since social bots are designed to mimic human behavior and interact with other users in social networks, their activities can also be represented as nodes and edges in a graph [53]. Graph-based approaches for social bot detection involve analyzing the patterns of interactions between nodes, such as the frequency and type of interactions, the timing and duration of interactions, and the structural properties of the graph. These methodologies can be employed to discern dubious behavioral patterns that exhibit characteristics of social bots, including but not limited to, excessive automation, recurrent actions, and anomalous network configurations.

Cornelissen et al. [61] merged network formation metrics with UML techniques in their study. The proposition is that utilizing a K-1 graph would be more efficient in discerning between bots and humans on social media than utilizing K-2 graphs. This was challenged by the examination of a Twitter dataset. Hurtado et al. [62] put forth the idea that accounts exhibiting similar temporal patterns and high connectivity are likely to be bots. Temporal and network data were employed to identify political bots on Reddit. In [63], BotCamp was introduced as a system that created graph structures based on user social behavior and utilized this information to cluster bots and detect social bot groups. Similarities among bots were identified in a campaign, such as temporal correlation, sentimental alignment, and topical grouping. They also observed that bots sometimes vied for human attention and engaged in debates, which they categorized as either positive or negative interactions. To identify new bot interactions in social campaigns, the authors devised an automated interaction classifier.

Generally, graph-based approaches have shown promise in detecting social bots, as they can capture the complex patterns of behavior that are characteristic of these entities. However, they also have limitations, such as the need for high-quality data and the potential for false positives and negatives. As social bots continue to evolve and become more sophisticated, graph-based approaches will need to adapt and improve to stay effective.

### 2.2.3 Anomaly-Based Bot Detection

This type of detection method relies on the idea that regular users of social media networks have no incentive to engage in unusual behaviors that do not provide any benefits. Therefore, if a user displays strange behavior while encountering certain restrictions, it is highly likely that the user is malicious. When abnormal behavior is detected within a group of users, they are categorized as a bot. This technique is commonly used in the initial stage of bot detection, which involves constructing reliable datasets and extracting information from identified bots to enhance future detection methods[56].

Identifying bots on Twitter is a challenging task due to the need for extensive datasets to train detection models, the bots' capability to adjust and avoid detection, and the intricacy of representing the varied characteristics of Twitter networks. As a solution, a group of researchers [64] considered the bot detection problem as an anomaly detection issue and developed a new model named anomalies detected in Twitter-attributed networks (ADNET). ADNET employs minimal labeled data to identify anomalies in Twitter networks by dividing the network topologically and picking the most informative nodes in each section. These nodes are used to train an auto-encoder that learns graph representations and reconstructs nodal attributes. The errors created by the auto-encoder are employed to score and rank nodes, which are then included in the labeled subset of the network until the stopping criterion is

achieved, leading to efficient bot detection on Twitter networks with limited labeled data.

Echeveria and Zhou [65] discovered a group of Twitter users with unusual behavior in tweet locations, such as quoting passages from a novel at random. They chose six million English-speaking Twitter users at random and categorized them as social bots. Likewise, the DeBot model was suggested by Chavoshi et al. [44] for monitoring the pertinent behaviors of specific users, and it was concluded that users who posted 40 or more tweets in an hour or had very similar behavior were social bots. The theory was that humans would not have numerous particular relevant activities over an extended period.

One of the main problems when using an anomaly-based approach for Twitter bot detection is the difficulty in defining what constitutes "normal" behavior on Twitter. Twitter users can have a wide range of behaviors and actions, making it challenging to determine which behaviors are unusual enough to be considered anomalous. Additionally, bots can adapt and change their behavior over time, making it difficult to create a static model that can detect them accurately. Finally, there is always the risk of false positives, where legitimate users are incorrectly flagged as bots, which can have negative consequences for these users.

### 2.2.4 ML-Based Bot Detection

ML is a field that uses algorithms and data to imitate human learning and gradually improve accuracy over time [66]. It is a cost-effective and efficient approach to tackling big data-related problems [56]. It is defined as "an area of research that enables computers to learn independently without the need for explicit instructions to be programmed" [67]. When there is ambiguity about what to look for in the data, such as determining factors that influence a movie-goer's preference

or dislike of a movie, ML is particularly useful. Consequently, ML algorithms have successfully responded to the "Netflix challenge," which involves predicting movie ratings based on user responses [68]. However, in situations where data mining goals can be described more directly, ML has been less successful. For example, WhizBang's effort to utilize ML to find individuals' resumes on the internet was an intriguing yet ultimately ineffective instance [68]. Table 2.2 presents an exhaustive list of advantages, disadvantages, and appropriate use cases for each technique.

**Table 2.2: General Overview Of Bots Detection Technique**

| Detection Technique | Advantages | Disadvantages | Appropriate Use |
|---|---|---|---|
| **Crowdsourcing-Based** | - Cost-effective<br>- Can leverage human intelligence<br>- Can provide diverse perspectives | - Prone to bias<br>- Can be time-consuming<br>- May require a large sample size | - Ideal for small-scale detection<br>- Effective for identifying simple bots with easily observable patterns |
| **Graph-Based** | - Can detect complex bot networks<br>- Can reveal social interactions and behaviors<br>- Can identify botmasters | - Requires prior knowledge of the network structure<br>- Can miss anomalies or outliers<br>- May require substantial computational resources | - Ideal for detecting sophisticated bots<br>- Effective for identifying bot clusters and networks |
| **Anomaly-Based** | - Can detect novel and unknown bots<br>- Does not require prior knowledge of bot behavior<br>- Can detect subtle deviations from normal patterns | - Can produce false positives<br>- Can be computationally expensive<br>- Can be vulnerable to adaptive bots | - Ideal for detecting emerging bot patterns<br>- Effective for identifying bots with non-obvious patterns or behaviors |
| **ML-Based** | - High accuracy<br>- Can learn from large and complex datasets<br>- Can identify multiple features and behaviors simultaneously | - Requires significant expertise and resources<br>- Can produce biased results | - Ideal for large-scale detection<br>- Effective for identifying a wide range of bot patterns and behaviors |

The field of ML has three primary branches, namely supervised, semi-supervised, and unsupervised techniques, which are closely associated with data prediction.

- Supervised: Supervised machine learning is a type of ML in which an algorithm is taught using a labeled dataset. Labeled data refers to input data that has already been associated with the correct output or target variable. The goal of supervised ML is to use this labeled dataset to learn a mapping amongst the input and output variables to the algorithm that can predict the correct output variable for new, unseen inputs [69]. Supervised ML using a training set and a test set is a common approach to evaluating the performance of a supervised learning algorithm. The training set is usually a random subset of labeled data, accounting for 70-80% of the total, with the remainder designated as the test set. The supervised architecture for training and testing is depicted in Figure 2.1.

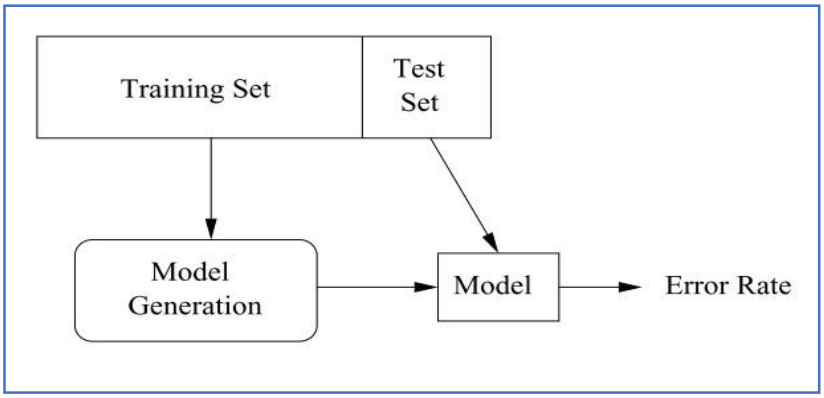



**Figure 2.1: Training and Test Sets in Supervised Model** [68]

- Unsupervised: The research proposes that while supervised ML is an effective approach for detecting Twitter bots, it has some limitations, such as the dependence on labeled data and difficulty in accurately classifying bots that mimic human behavior. To overcome these

limitations, an alternative approach using unsupervised learning methods is proposed. This approach clusters accounts based on their similarities and identifies commonalities between groups of accounts without requiring labeled data [54], [56]. Figure 2.2 illustrates the difference between supervised and unsupervised learning. In Figure 2.3a, the input data is labeled with known classes, and a supervised classification algorithm is used to categorize the data into those known classes. In contrast, in Figure 2.3b, an unsupervised clustering technique is utilized to understand the structure of the data, without any known classes.

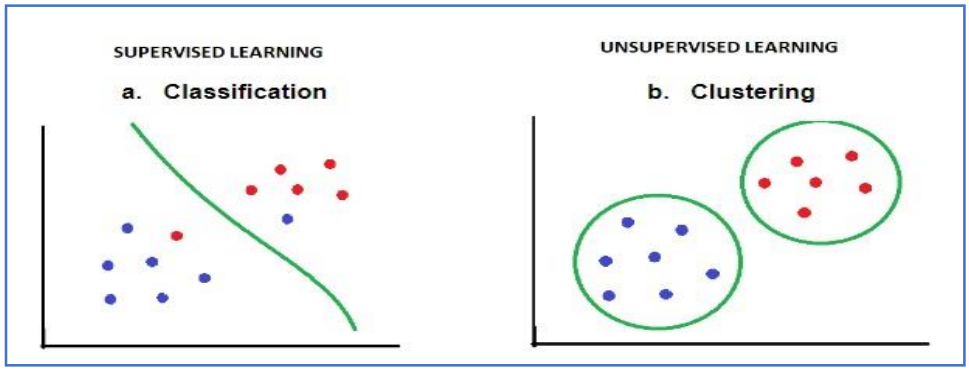



**Figure 2.2: Supervised Classification VS Unsupervised Clustering** [70]

- Semi-supervised is an ML technique that utilizes both labeled and unlabeled data to construct a model. A strategy for this technique involves utilizing labeled data to generate class models and utilizing unlabeled data to refine class boundaries, especially in a binary classification problem where positive examples belong to one class and negative examples belong to the other [54]. Figure 2.3 demonstrates that using both labeled and unlabeled examples can refine the decision boundary, as shown by the solid line. This approach can also identify outliers or noise in the labeled data [69].

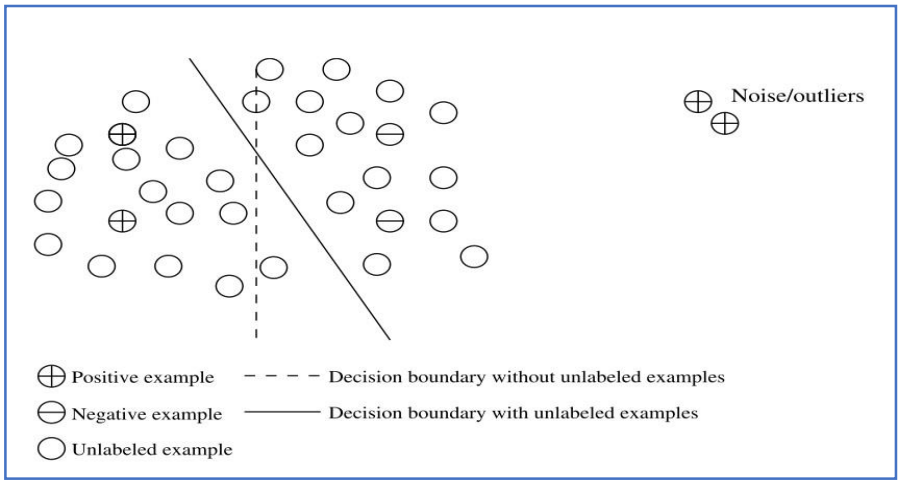



**Figure 2.3: Semi-supervised Learning** [69]

## 2.3 Clustering

The three types of clusters used in this research are well-separated, prototype-based, and density-based [6].

### 2.3.1 Well-Separated Clustering (WSC)

WSCs are those that have a clear and distinct separation between them. This type of clustering is useful when the data objects can be easily classified into distinct, non-overlapping clusters based on a set of features or characteristics. [71][72]. Figure 2.4(a) illustrates two distinct point sets in a 2-dimensional space that form well-separated clusters. These clusters have a greater distance between any 2- points from dissimilar groups compared to the distance between any two points within the same group. It is important to note that well-separated clusters can take on various shapes and do not necessarily need to be spherical [6].

### 2.3.2 Prototype-Based Clustering

Prototype-based clustering, also known as centroid-based clustering, involves identifying a set of "prototype" points that represent the clusters in the dataset [73]. This type of clustering is useful when the data points are more spread out or when

there is an overlap between clusters [74]. The k-means algorithm is a popular method for prototype-based clustering that categorizes data points into clusters by comparing their resemblance to each prototype cluster [73]. An instance of clusters based on central points is demonstrated in Figure 2.4(b) [6].

### 2.3.3 Graph-Based Clustering

Graph-Based Clustering also so-called Contiguity-Based Clustering involves identifying clusters based on the structure of a graph that represents the relationships between data points [75]. This type of clustering is useful when the data points can be expressed as graph nods and the connections between them can be captured as edges [76]. In other words, a collection of items that are linked together exclusively within the set and are not associated with any items beyond the set. Consequently, the contiguity-based clustering approach mandates that every element within a given cluster must maintain closer proximity to at least one other constituent in the same cluster than to any constituent belonging to a separate cluster [77], [78]. An illustration of this type of clustering for 2-dimensional is provided in Figure 2.4 (c). This clustering definition proves beneficial when dealing with intricate and non-uniform clusters. However, this method may encounter difficulties when attempting to identify clusters in the presence of noise. For instance, as demonstrated by the presence of two spheral shapes of clusters in Figure 2.4 (c), a slender bridge composed of points can combine two separate clusters [6].

### 2.3.4 Density Clustering Based

It involves identifying dense regions of data points that are encompassed by areas of low density and is particularly useful for datasets that contain noise or outliers and where data points are not separated. DBSCAN is often used for density clustering, focusing on high-density areas and disregarding unclustered data

points[79]. In Figure 2.4 (d), some density clusters are shown for a dataset that includes noise. The circular clusters in Figure 2.4 (c) are not combined in Figure 2.4 (d) since the connection between them merges with the noise, and the curve depicted in Figure 2.4 (c) does not constitute a cluster in Figure 2.4 (d) because of the interference from noise. Density clustering is ideal for complex, tangled clusters with outliers and noise, as contiguity-based clustering may not handle such scenarios well due to noise potentially connecting clusters. [6], [79], [80].

The clustering algorithm and definition significantly impact analysis results. Considering goals and data is vital when choosing an approach, as it reveals valuable insights into dataset structures using diverse cluster types.

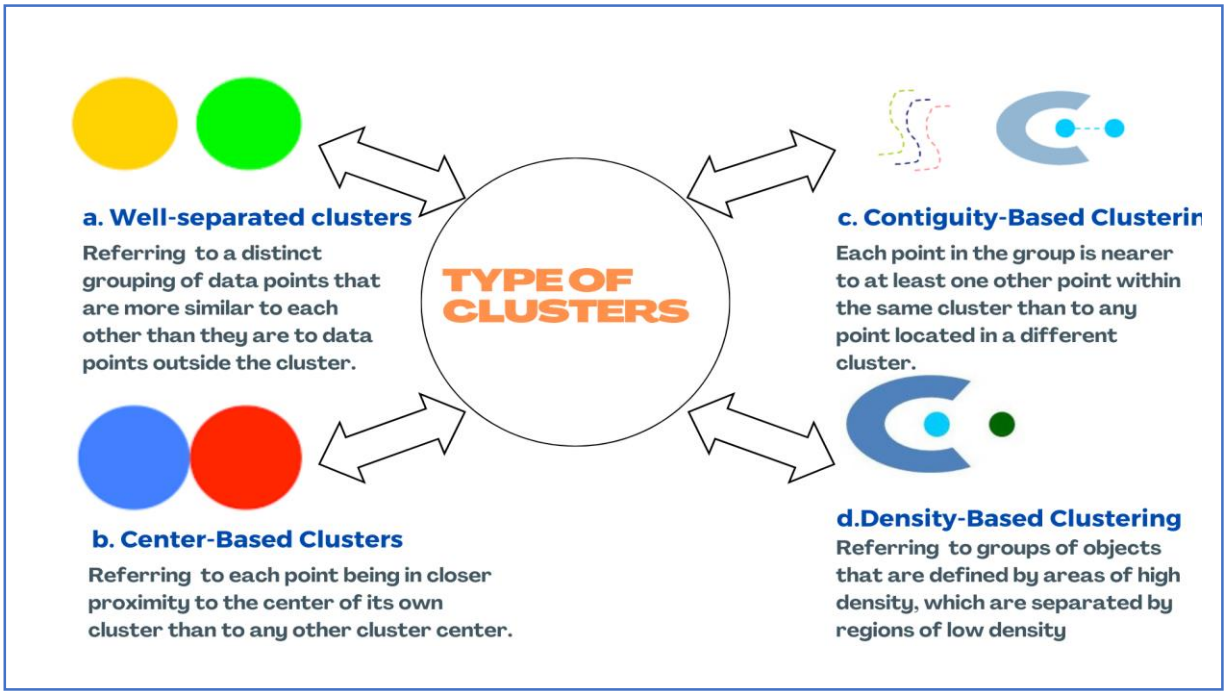



**Figure 2.4: Types of Clusters**

## 2.4 Distance Measure

Distance metrics are utilized in multiple domains, such as machine learning, statistics, data mining, and information retrieval, to quantify the resemblance or contrast between two entities or data points. In essence, distance measures quantify

the distance that separates two points in a multidimensional space [68]. There is a wide range of distance measures available for data clustering, but the most commonly used ones are metric functions, including EDs, the MD, the MKD, and the CBD. In addition to these metrics, the Jaccard index and Cosine Similarity are also frequently employed distance measures [81]–[83].

### 2.4.1 Distance Measure Properties

The theory of distance measures is based on a set of properties that a distance measure should satisfy to be useful and valid. The properties listed in Table 2.3 are fundamental to the theory of distance measures, as they establish the criteria for meaningful, consistent, and valid distances. These properties form the basis for defining and evaluating a wide range of distance measures, including well-known ones like ED, MD, and Hamming distance. When choosing a distance measure, it is crucial to consider the unique requirements of the application and the inherent characteristics of the data being analyzed.

**Table 2.3: Key Principles of Distance Measures: Validity and Coherence**

| Property | Description | Equation |
|---|---|---|
| **Non-negativity** | A distance measure should always be non-negative, meaning that the distance between any two points cannot be negative. This property ensures that distances are always meaningful and consistent with the underlying space [68], [84]. | Dis (x, y) ≥ 0 |
| **Symmetry** | A distance measure should be symmetric, meaning that the distance between x and y is the equivalent of the distance between y and x. This property ensures that distances are not affected by the order in which the points are considered [68], [84]. | Dis (x, y) = dis (y, x) |
| **Triangle-inequality** | A distance measure must follow the triangle inequality, guaranteeing that the distance from x to z via y is always shorter or equal to the sum of the distances from x to y and y to z. This maintains consistency with the underlying space and preserves geometric laws. [68], [84]. | Dis (x, z) ≤ dis (x, y) + dis (y, z) |
| **Positivity** | A distance measure should always be positive, except the distance between a point and itself, which is zero. This property ensures that distances are meaningful and non-trivial [68], [84]. | Dis (x, y) > 0, for x ≠ y<br>Dis (x, x) = 0 |
| | | |

### 2.4.2 Euclidean Distance (ED)

Euclidean Distance (ED) is a commonly used measure of similarity between point pairs in an N-dimensional space. A definition of it is root mean square variations between the corresponding coordinates of two points. Mathematically, if we have two points in n-dimensional space, that are represented as vectors x and y, then the ED between them is given by Equation 2.1 [6]:

$$\mathbf{d}(\mathbf{x}.\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} \quad \textbf{(2.1)}$$

where xi, and yi, are the coordinates of the two points along each of the n dimensions.

In ML, ED is commonly used to measure similarity between samples and as a distance metric for clustering algorithms like K-means. It is widely applicable in various domains, including computer vision, natural language processing, bioinformatics, and related fields.

### 2.4.3 Manhattan Distances (MD)

The MD, also called the taxicab distance or L1 distance, provides a means to calculate the separation between two points in a grid-like arrangement. It is termed MD due to its similarity to the navigation path that a taxi would follow in a city with a rectangular grid of streets [68]. The formula to calculate MD between two points (x1, y1) and (x2, y2) is illustrated in Equation 2.2.

$$\mathbf{d}(\mathbf{x}.\mathbf{y}) = |x_2 - x_1| + |y_2 - y_1| \quad \textbf{(2.2)}$$

where "|" denotes absolute value.

MD and ED are distance metrics used in clustering algorithms to measure similarity between data points. MD sums absolute differences between coordinates, while ED calculates root mean square differences. MD is ideal for high-dimensional data with numerous irrelevant features as it assigns equal weight to all dimensions[85], [86], whereas ED is better suited for low-dimensional data where the difference between features is significant.

### 2.4.4 Minkowski Distance (MKD)

MKD is a distance measure utilized in machine learning to evaluate the likeness or unlikeness between two points in a multi-dimensional space. It is a generalization of both the ED and the MD, the formula for the MKD between two variables x, and y is defined as in Equation 2.3 [6]:

$$\mathbf{d}(\mathbf{x}.\mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \quad \textbf{(2.3)}$$

where p = 1 is equivalent to the MD and p = 2 is equivalent to the ED.

The choice of the best distance metric for the clustering technique is based on the data category being clustered and the clustering algorithm being used. In general, ED is best for continuous variables with no outliers, while MD is better suited for variables with high dimensionality or when there are outliers in the data. MKD can be used in both cases, and the choice of the parameter p can be tuned to balance between the two [87]–[89].

### 2.4.5 Canberra Distance (CBD)

The CBD is a metric used to measure the distance for twain points in an N-dimensional space. It is a weighted version of the MD, which takes into account the magnitudes of the values in each coordinate. Equation 2.4 is used for CBD [90]:

$$\mathbf{d}(\mathbf{x}.\mathbf{y}) = \sum_{i=1}^{n} \frac{|xi - yi|}{|xi + yi|} \quad \textbf{(2.4)}$$

In this context, xi and yi refer to the ith values coordinate for x and y, respectively. The denominator in the equation normalizes the distances by taking into account the magnitudes of the values in each coordinate. The CBD is often used in clustering algorithms and classification problems due to its robustness to outliers. It is particularly useful in situations where the data contains outliers or when the magnitudes of the values in different dimensions vary widely[91].

The CBD is a popular distance metric in clustering algorithms, including fuzzy clustering and hierarchical clustering, as well as in classification, computer security, and spam detection systems. Its usefulness is particularly evident when dealing with datasets containing outliers or variations in values across different dimensions [91], [92]. Considering its potency in measuring distances between points in a multidimensional space, the CBD has become a preferred choice in many applications, including clustering. As a result, it is possible to use the CBD in the suggested system to calculate the distance between the centroid and each point through the k-means algorithm, instead of relying on EDs.

The CBD has many advantages such as robustness to outliers and handling data with varying magnitudes. However, it also has drawbacks to be considered. It is sensitive to zero values, leading to undefined distances when the denominator is zero. This can be problematic when datasets have many zeros, resulting in inaccuracies and unstable outcomes [91]–[94]. Computationally, the CBD is slower

than the ED due to additional operations like division and absolute value calculations, particularly for large or high-dimensional data [95]. It can yield biased results with high-dimensional data, as a single dimension may dominate the distance, causing misleading clustering [96].

It is important to consider the strengths of each distance metric and how they may impact the accuracy and performance of the clustering technique for a given problem. It is also crucial to note that the sensitivity of distance metrics to changes in the data should be taken into account when making a selection. Using the CBD instead of ED in clustering algorithms is beneficial in certain cases, particularly with count data or when balancing the importance of large and small distances. CBD considers both absolute differences and magnitudes of coordinates, which suits data with small values or count data. In contrast, ED only considers absolute differences and is sensitive to data scale and outliers, potentially affecting clustering accuracy.

## 2.5 K- Means Technique

K-means is a frequently used unsupervised clustering method that seeks to divide a dataset into K clusters, where K denotes a pre-established number of clusters [68], [80]. It is a clustering method that is prototype-based and partitional in nature, and its objective is to identify a predetermined number of clusters (K) through the use of centroids to represent them[6]. The following is an explanation of the technique of the K-means algorithm functions:

1) Initialization: Selecting K points at random from within the collection of data to serve as the starting centroids.
2) Allocate points to clusters: Assigning points to clusters involves computing the distance for each point and every centroid, then appointing each point to the cluster with the closest centroid.

3) Update the centroids: Computing the mean of all data points in each cluster and setting the cluster centroid to that mean value.
4) Reiterate steps 2 and 3 until convergence: Iterate through steps 2 and 3 until the clusters do not change anymore, or until the upper limit of repetitions is achieved.
5) Return: Final centroids represent the K clusters.

It should be noted that the algorithm might converge to a local minimum. Thus, it is frequently executed several times using different randomly chosen initial centroids to boost the likelihood of discovering the global minimum. Additionally, there are different variations of the K-means such as the K-medoids, which employs the medoid (i.e., the most centrally located point in a cluster) instead of the mean as the centroid, and the K-means++ technique, which enhances the initialization step by selecting initial centroids that are widely separated from each other.

### 2.5.1 The Importance of Initializing Clusters in K-Means

The selection of cluster centroids during the initialization stage is a critical aspect that can impact the performance of the K-Means algorithm. When the initial centroids are not appropriately chosen, the algorithm may produce suboptimal or invalid cluster solutions [68]. To address this issue, a new initialization method called K-Means++ was proposed [97]. This technique aims to achieve clustering results that are comparable to optimal clustering by maximizing the distance between the initial centroids. By doing so, the chance of selecting centroids that represent different clusters is increased, leading to more accurate cluster solutions. In [98], light was shed on the crucial role of proper initialization of cluster centers in the k-means method, highlighting both theoretical and practical aspects. As a non-robust algorithm, k-means are known to be sensitive to outliers, which can affect clustering quality on noisy data. Popular seeding methods like k-means++ may even

amplify this problem by selecting outliers in the worst-case scenario. To tackle this issue, it was suggested to use a mixture of $D^2$-sampling and uniform sampling to choose a subset of candidate centers, amounting to $O(k/\delta)$ centers, where $\delta$ ranges from 0 to 1. The empirical results demonstrate that the robust version of k -means++ outperforms other seeding methods, making it more resilient to outliers and producing more accurate clustering results.

In [68], a new method was introduced to enhance the efficiency of k-means++ clustering, which involves incorporating local search strategies. The study conducted experiments comparing the proposed algorithm's performance to that of the traditional k-means++ algorithm for k=50 and k=25 using various datasets. The findings reveal that the proposed algorithm outperforms the traditional k-means++ algorithm significantly, with cost reduction ranging from 8% to 35%.

Overall, while K-Means++ is an effective improvement over the standard K-Means algorithm, some disadvantages should be considered: Computationally Intensive, Sensitivity to Outliers, and K-Means++ are limited to using ED to calculate distances between data points and cluster centers. This may not be suitable for all types of data, such as categorical data or data with non-linear relationships.

Another technique appropriate to improve the initialization of K-Means is hierarchical clustering. This involves first clustering the data points using a hierarchical clustering algorithm and then using the resulting cluster centers as the initial centroids for K-Means. This can be particularly useful when dealing with large datasets or when the data is highly dimensional. Two effective hierarchical initialization techniques, named Variance Partitioning (Var-Part) and principal component analysis partitioning (PCA-Part), were developed by Su and Dy [99]. These approaches are linear, deterministic, and order-invariant, aimed at addressing the challenge of initializing the cluster centers. They overcome the limitations of

prior methods that had linear complexity, random initialization, order sensitivity, and non-reproducibility. Experimental results show that Var-Part and PCA-Part have comparable performance, with Var-Part occasionally performing better. In [100], a straightforward alteration was suggested to the techniques presented in [99] that leads to a considerable improvement in their effectiveness. Rather than assigning each point to one of the two subclusters based on whether its projection falls on the same side as the mean point, the authors proposed a modification. It was suggested that the points' projections onto the partitioning axis could be perceived as a discrete probability distribution, and this distribution could be accurately depicted by a histogram. In another research study, Lailil Muflikhah and colleagues [101] suggested a clustering technique for sequences that utilizes the Hierarchical k-Means clustering algorithm. This method aims to enhance the k-Means of establishing an initial cluster center based on the mean outcome of the hierarchical clustering approach. The experimental findings indicate that the suggested method outperforms the traditional k-Means clustering algorithm in terms of performance measures.

Hierarchical clustering can be a useful technique for improving the initialization of K-Means. However, it is important to consider its limitations, such as its potential computational expense. Additionally, the technique can be sensitive to the choice of distance metric used, as different metrics may lead to varying clustering results. Similarly, the choice of the linkage method can also impact the clustering outcome. Lastly, specifying the cluster numbers in advance can be challenging, particularly when dealing with high-dimensional data.

### 2.5.2 Choosing the Best K Parameter

Determining the appropriate number of clusters, also known as K, is a challenge in clustering [102]. This issue arises not only in hierarchical and density-

based clustering but also in general clustering. There are no fixed rules to decide the cluster numbers, and several methods exist that work only in specific cases. The Elbow technique and the Silhouette technique are two simple techniques for determining cluster numbers. These methods generate graphs that offer a rough estimate of the number of clusters that should be used in clustering [89]. The choice of distance metric is a critical factor in creating good clusters and is also important when using the Elbow and Silhouette methods to decide the optimal cluster numbers [89]. The ED metric is the default choice, but other metrics such as Manhattan and Minkowski can also be used.

The elbow technique decides the ideal cluster numbers in k-means clustering by plotting the cost function's value against various numerals of k. As shown in Figure 2.5 when k rises, the average distortion decreases, and instances become closer to their respective centroids, while the number of instances in each cluster decreases. However, the enhancements in average distortion diminish as k continues to increase. The elbow point represents the stage where the reduction in distortion improvement is most significant, indicating that further dividing the data into additional clusters is not necessary. [89], [103]–[106].While the Elbow technique is a popular and straightforward technique for deciding the optimum number of clusters in K-means clustering, it has some limitations that must be taken into account.

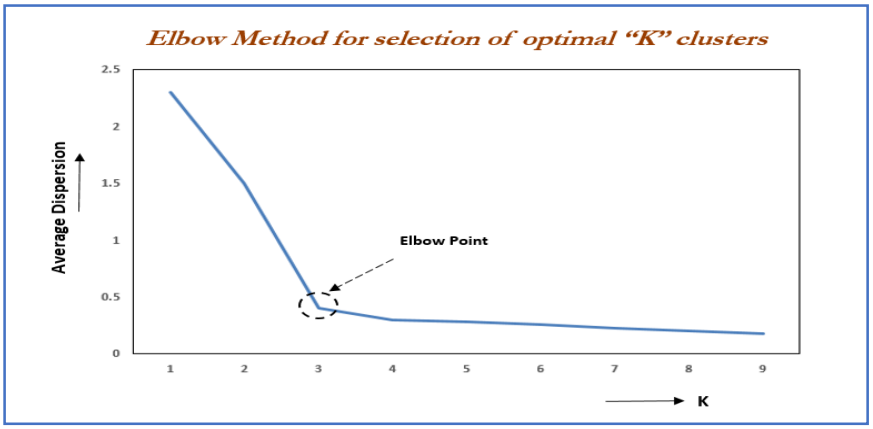


**Figure 2.5: Elbow Technique for Identifying Cluster Numbers** [107]

The Elbow technique has limitations when it is applied to complex datasets with non-uniform cluster sizes and shapes. In such cases, the resulting plots may lack a clear elbow or have an elbow point that does not represent the optimal number of clusters. Relying solely on the Elbow method may not yield the best results [108]. Additionally, the Elbow method assumes a spherical data distribution and clusters of similar size and density. However, real-world scenarios often deviate from this assumption, leading to suboptimal clusters. Therefore, alternative clustering algorithms and techniques should be considered for non-spherical and heterogeneous data [109].

Silhouette metric is a technique employed to assess the clustering outcomes' efficiency in unsupervised learning algorithms such as k-means. The method assesses the similarity of data points within clusters and the dissimilarity between clusters. The value of silhouette scope is from -1 to 1, where a value closer to 1 indicates a better clustering result. The silhouette method can be used in k-means to determine the optimal cluster numbers for a given dataset. In Figure 2.6 the method involves plotting the average value of the silhouette for various values of k and picking the k value that results in the topmost mean silhouette score [103], [110]–[113].

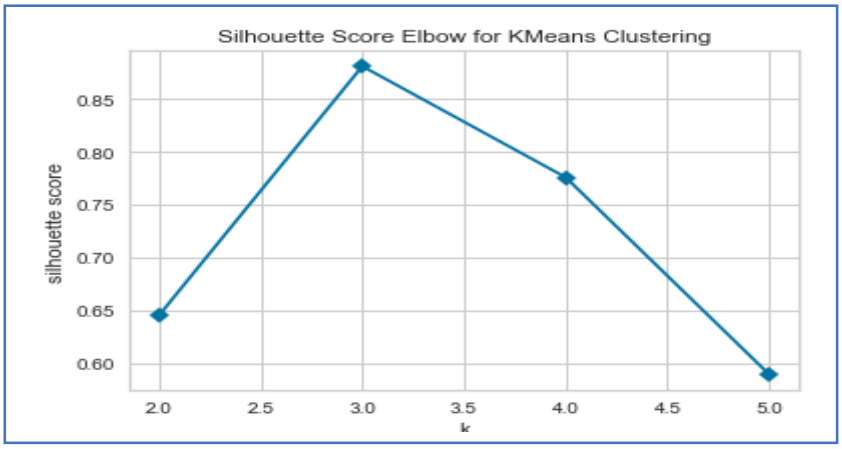


**Figure 2.6: Optimal K value based on Silhouette Coefficient**

While silhouette is a popular technique for defining the optimal cluster numbers in k-means clustering, it has some limitations that should be considered. Firstly, the silhouette method may not be suitable for datasets where the clusters are not well-separated or where the data is of high dimensionality. In such cases, the WCSS may not decrease significantly with increasing values of K, making it difficult to determine the silhouette point. The silhouette method assumes WCSS reflects clustering quality, but it may not hold for all datasets due to factors like data distribution and outliers.

## 2.6 DBSCAN Algorithm for Grouping Data Points

DBSCAN is a clustering technique that clutches points that are close to each other in dense regions and separates points that lie in low-density areas [6]. The algorithm is capable of dealing with datasets that have different shapes and sizes and doesn't need any pre-existing information about the number of clusters [6]. It identifies "core points" which have at least "min_samples" neighboring points within a specified distance "eps", and expands clusters around these core points by recursively adding neighboring points. DBSCAN also identifies "noise points" that do not belong to any cluster [114]. A border point is a data point that is within a core point distance but doesn't have enough nearby data points to be considered a core point itself. It's on the edge of a cluster and is considered part of the cluster, but not as strongly as core points as illustrated in Figure 2.7. The DBSCAN algorithm requires two parameters, epsilon (eps) and minimum points (min_samples), which need to be set before running the algorithm.

1) The parameter epsilon sets a limit on the distance between two points, beyond which they are not considered to belong to the same cluster.

Points within this distance are said to be "dense". The larger the epsilon value, the more points will be included in a cluster, but it may also cause different clusters to merge into one. On the other hand, a smaller epsilon value may break a single cluster into multiple smaller clusters. Thus, epsilon must be set carefully based on the underlying data distribution and the desired clustering output.

2) The parameter minimum points (MinPts) specifies the minimum points number desired to form a dense region, called a "core point". Core points are the starting point for a cluster and all points located within a range of epsilon distance from a core point are considered part of the same cluster. Setting a higher value for min_samples will result in more conservative clustering, meaning that only points in denser regions will be included in clusters. Conversely, setting a lower value for min_samples will result in more aggressive clustering, including more points in the cluster, including some points in lower-density regions.

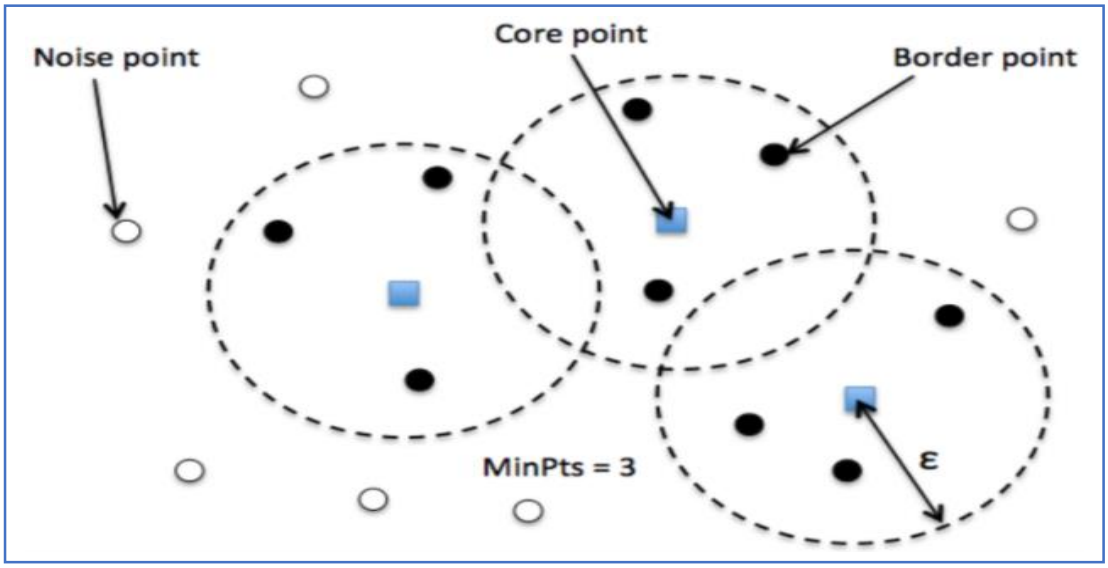



**Figure 2.7: DBSCAN's Categories of Data Points: Core, Border, and Noise Explained**

### 2.6.1 Understanding the Inner Working of the DBSCAN Algorithm

To begin with the algorithm, a point is chosen randomly and uniformly from the dataset. This point is then checked to see if it is a core point, meaning it has at least minPts number of points in its epsilon neighborhood. The algorithm then proceeds to identify the connected components of all core points, ignoring noise points. Any non-core point is then assigned to the nearest cluster if it is within the epsilon neighborhood, or marked as noise otherwise. The algorithm continues this process, exploring all points and classifying them as core, border, or noise points. Once all points have been explored and classified, the algorithm stops.

To implement DBSCAN, the algorithm requires two parameters: epsilon (ε), and MinPts, the algorithm then proceeds as follows [115]:

1) Randomly select a point from the dataset.
2) Identify all points within distance ε of the selected point.
3) If there are less than MinPts located within the neighborhood, then the point should be labeled as noise and the program should proceed to the next point.
4) If there are at least minPts points within the ε neighborhood, create a new cluster and add the selected point and all the neighboring points to the cluster.
5) Expand the cluster by adding all points within ε of any point in the cluster.
6) Repeat the process for all unvisited points until all points have been visited.
7) The resulting clusters may have different shapes and sizes, as they are determined by the density of the points rather than any pre-defined shape or structure. Additionally, points that do not belong to any cluster are identified as noise. Figure 2.8 presents a comparison of how

clustering algorithms, specifically K-means, and DBSCAN, operate differently [116].

Overall, the choice of epsilon and min_samples parameters plays a crucial role in the DBSCAN algorithm's performance and can significantly impact the clustering output. It is essential to understand the underlying data distribution and how it relates to the chosen parameter values to ensure optimal clustering results.

When selecting the MinPts value for DBSCAN clustering, it is recommended to use domain knowledge and familiarity with the data set. Several rules of thumb can help in selecting an appropriate value for MinPts, such as choosing a larger value for larger or noisier data sets, setting MinPts greater than or equal to the dimensionality of the data set, and using the default value of MinPts=4 for 2-dimensional data [117]. For data sets with more than two dimensions, the recommended MinPts value is 2 times the dimension numbers in the dataset [118]. However, it is important to note that these guidelines are not absolute, and the optimal value of MinPts may still rely on the particular characteristics of the dataset [119]–[121].

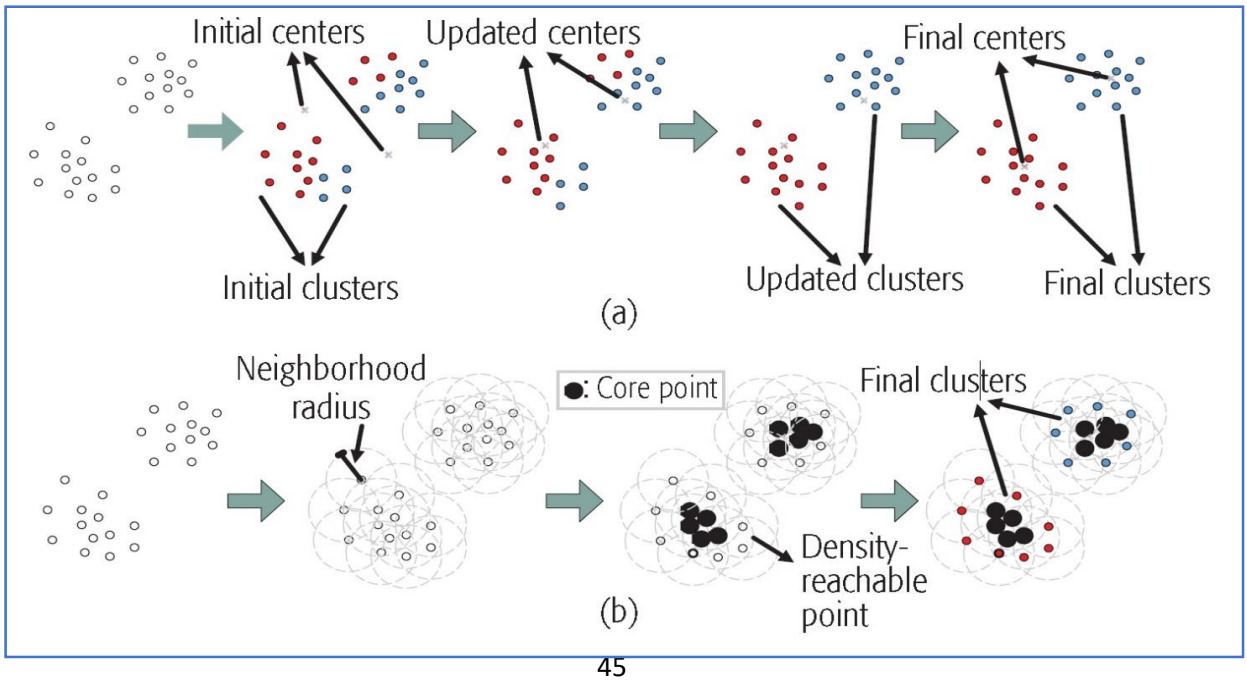


**Figure 2.8: K-means VS DBSCAN Algorithms** [116]

## 2.7 k-Nearest Neighbor (KNN)

The KNN technique is a popular ML technique used for classification and regression tasks. The traditional supervised KNN algorithm is trained on labeled data, where each example has a known class or output value [6]. However, unsupervised KNN does not require labeled data and seeks to identify natural clusters in the data using a similarity metric, such as Euclidean, Mahalanobis, or cosine similarity [122], [123]. The unsupervised KNN algorithm works by selecting a value for k, which determines the neighbor's numbers to consider, and finds the k-nearest neighbors for each data point, assigning it to the same cluster as its nearest neighbors. This process is repeated until all data points have been allotted to a cluster [124], [125]. Unsupervised KNN is advantageous in identifying natural patterns in the data without prior knowledge of class or output values. However, it requires an appropriate selection of k and can be sensitive to the choice of similarity metric and other parameters. Additionally, it may not be suitable for high-dimensional or sparse datasets. Figure 2.9 provides a visual aid for understanding the functioning of k-Nearest Neighbors [126].

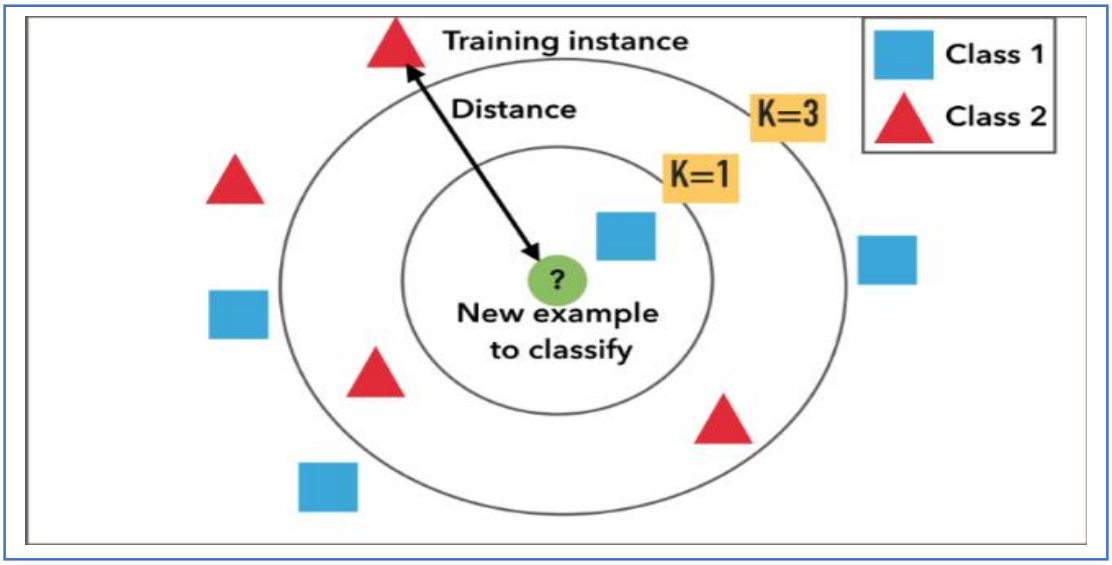


**Figure 2.9: Visualizing k-Nearest Neighbors (k-NN) Algorithm**

### 2.7.1 Implementation Methods of kNN

There are different methods for implementing kNN, including the ball tree, brute force, and k-d tree. A detailed discussion of every method will be presented here:

1) Brute Force Technique: it is a direct approach for implementing kNN that involves measuring the distance between each training instance and the test instance, then choosing the k closest neighbors using the distance measure. While this method is uncomplicated to implement and does not require any pre-processing of data, it can be computationally intensive and impractical for large datasets since it involves computing distances between all pairs of instances [127]–[129].
2) k-d Tree Method: It is a data structure that partitions the feature space into smaller regions, making it easier to search for the k nearest neighbors. It recursively splits the feature space into two halves along the median value of a chosen feature until all instances are separated into individual leaves. During the process of finding the k closest neighbors, the method only traverses the regions that are likely to contain the nearest neighbors, making it more efficient than the brute force method. This method is suitable for high-dimensional datasets, but it requires preprocessing the data to construct the k-d tree, which can take some time [130]–[132].
3) Ball Tree Method: The ball tree method is another data structure that partitions the feature space into smaller regions. It does this by creating a ball around each training instance with a radius that encompasses the k nearest neighbors

[129]. The algorithm then creates a tree structure where each node corresponds to a ball and the children nodes represent the two sub-balls created by splitting the parent ball. During the process of finding the k closest neighbors, the algorithm only traverses the regions where the balls overlap, making it more efficient than the brute force method [133], [134]. This method is also suitable for high-dimensional datasets and requires preprocessing the data to construct the ball tree [135].

In summary, the choice of which kNN method to use depends on the characteristics of the dataset, such as its size and dimensionality. The brute force method is simple to implement but can be slow for large datasets. The k-d tree and ball tree approaches are more efficient for high-dimensional datasets but require preprocessing the data to construct the trees.

### 2.7.2 Selecting Appropriate Parameter of k

Selecting an appropriate value of the k parameter is crucial for achieving optimal performance in unsupervised kNN algorithms. The k value determines the neighbor's number considered for classification or clustering. Selecting too small a value for k can result in overfitting while choosing too large a value can lead to underfitting. Therefore, the suitable choice of the k value must be based on the dataset's properties and the analysis goals. In general, selecting the optimal k parameter involves balancing the tradeoff between the accuracy of the model and its complexity. Various techniques, such as cross-validation and grid search, can be employed to identify the optimal k value for a given data set. Ultimately, selecting the suitable k value is crucial for the success of unsupervised kNN algorithms and can have a significant influence on the quality of the results obtained.

According to [60], selecting the ideal value of K in kNN is a crucial decision that necessitates thoughtful deliberation. The commonly employed techniques to determine the optimal k value include cross-validation, domain knowledge, and trial and error. Nonetheless, the most effective method depends on the specific nature of the problem and the data being examined. The research paper cited in the source indicates that cross-validation is a dependable technique for identifying the optimal k value for a given problem [136]. The cross-validation approach requires significant computational resources and may yield multiple values of k.

The proposed method in [137] for determining the optimal k value for the k-NN algorithm involves using a local complexity approach to construct a dataset profile, creating a feature vector based on metrics such as mean, median, midrange, and skewness, and computing an optimal k value for each historical dataset. The feature vector and corresponding k value are stored as metadata, and a prediction model is built using this metadata to predict the optimal k value for new datasets. The experimental outcomes prove the efficiency of the suggested approach, but details about the size, nature, and source of the dataset(s) used in the experiments are not provided, which limits the assessment of the method's generalizability.

The DC-LAKNN algorithm proposed in [138] is a new approach to kNN-based classification that takes into account the second most frequent class and dynamically adjusts the value of k based on the quantity and distribution of data. The algorithm selects discriminatory classes at different k values to fine-tune the value of k. However, the trial and error method used to determine the k value may not always lead to the optimal result, and it can be time-consuming. Comparing the performance of DC-LAKNN with other kNN-based algorithms would help evaluate its effectiveness.

According to Lall and Sharama [139], for datasets with a sample size greater than 100, the optimal k value should be such that $k = \sqrt{N}$. However, it has been shown that this setting may not be suitable for all datasets [140]. The problem of selecting the appropriate value of k can be viewed as a model selection problem according to the Bayesian method, as noted in studies [141]–[143].

In summary, the ideal K value in kNN depends on the dataset's characteristics and problem. To strike a balance between overfitting and underfitting, consider the bias-variance trade-off. A smaller K makes the model flexible but prone to overfitting, while a larger K offers stability but risks underfitting. Choosing K often involves Bayesian methods, trial-and-error, and careful analysis, as there's no one-size-fits-all approach. Tailor your K choice to the specific dataset and problem at hand.

## 2.8 Dimensionality Curse

The Dimensionality Curse (DC) is a term used to describe a common challenge in ML and data analysis when dealing with high-dimensional datasets. The term refers to the fact that the complexity of the data increases exponentially with the number of dimensions, which can make it difficult to process, analyze, and visualize the data [68]. Table 2.4 outlines the major issues encountered when dealing with data that have a large number of dimensions, which are commonly referred to as the challenges of high-dimensional data. To mitigate the DC, various techniques can be used, such as dimensionality reduction, FS, and regularization methods. These techniques can help decrease the difficulty of the data and make it more manageable for analysis.

**Table 2.4: Challenges of High-Dimensional Data**

| Challenges of High-Dimensional Data | Description | Ref. |
|---|---|---|
| **Increased sparsity** | When dimensions in a dataset increase, it becomes more difficult to detect patterns and correlations between variables because the data becomes more sparse. | [6], [144] |
| **The distance problem** | In high-dimensional spaces, almost all pairs of points are equidistant, making it difficult to use distance-based metrics for comparison or identifying outliers. | [6], [145] |
| **Almost orthogonality problem** | In high-dimensional spaces, most vectors are nearly orthogonal, making it harder to identify meaningful patterns in the data. | [144] |
| **Overfitting** | With increasing dimensions, there is a higher risk of overfitting, where ML models may become overly complex and fit noise instead of underlying patterns. | [6], [146] |

## 2.9 Categorizing Features for Twitter Bots Detection

When distinguishing a bot and human accounts on Twitter, researchers often consider the features that differentiate them. For instance, some researchers use screen names and post locations to identify bot accounts [65], [147]. Nonetheless, individuals who create bots can readily devise ones that can evade detection by these types of models. To address this, many researchers use a predefined attributes list and accounts labeled set to train supervised ML models that determine thresholds for feature values, aiding in identifying bot accounts and estimating botnets.

It is crucial to select appropriate features for identifying Twitter bots, particularly with ML methods. One of the primary difficulties in this domain is the constantly changing capabilities of bots, leading to modifications in the distinct feature values and making them difficult to detect. Therefore, discovering resilient features is a current area of focus. The inquiry pertains to the number of sets of features that should be chosen to assist ML methods in detecting malicious Twitter bots. High-dimensional is prevalent in various fields, including SM, but it has several drawbacks such as excessive computation costs, overfitting, and poor overall

performance [148]. Getting rid of unneeded and repetitive attributes can aid in decreasing overfitting, conserving computing resources, and improving precision. Figure 2.10 presents a taxonomy of FS and extraction methods, along with the number of reviewed articles for each category. The taxonomy comprises three main categories: Metadata features, Tweet text features, and Merged features category.

## 2.10 Metadata Features

Metadata in Twitter refers to additional information about a tweet, including the date and time of creation, the user who posted it, the location, and any hashtags or mentions included in the tweet. Twitter metadata can also include information about the engagement and reach of a tweet, such as the number of retweets, likes, and replies, as well as the impressions and clicks it has received [149]. In Figure 2.10, Twitter metadata features are presented in two categories: FS and FE. The upcoming section will provide a detailed discussion of these different parts.

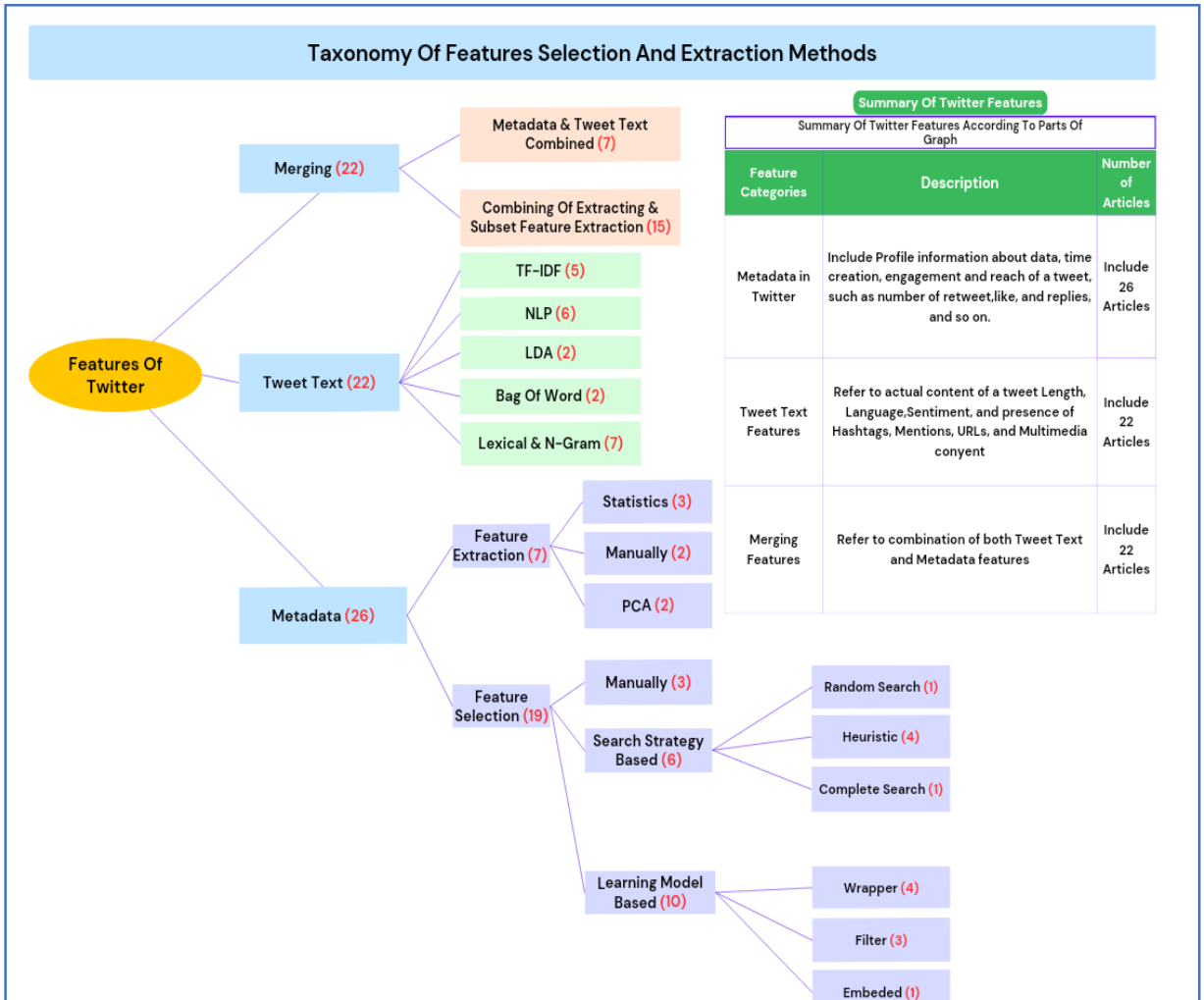


**Figure 2.10: Taxonomy of FS and FE Approaches based on Twitter Features**

### 2.10.1 Features Extraction Category

FE is a technique employed to mitigate dimensionality and enhance learning accuracy. This technique encompasses two categories of algorithms, which are linear and nonlinear techniques. Nonetheless, the principal component analysis (PCA) [150], [151], statistical methods [152]–[154], linear discriminant analysis (LDA) [46], and manual extraction [155], [156] are considered the most efficacious feature extraction-based dimensionality reduction methods.

In [150], the utilization of supervised ML methods and an enhanced SVM is explained to identify deceitful accounts on online social platforms and produce a model that can precisely portray counterfeit profiles. The resulting method achieved a 90% accuracy rate, surpassing the performance of the Nave Bayes (NB) and SVM algorithms, both attained 77.4% and 77.3%, in that order. In [151], the authors employed a robust set of features constructed through a combination of linear regression and PCA to detect Twitter spammers. The performance evaluation of the newly developed set of features demonstrated an improvement in accuracy, along with a low false-positive degree. In [154], a multi-objective hybrid approach was utilized to determine the optimal feature subset for identifying fraudulent Twitter profiles. [152] used an ensemble-learning approach to evaluate COVID-19 tweets. The method categorized data into credible and non-credible using 26 tweet- and user-level features. Tests showed the framework accurately identified COVID-19 tweets. In [153], we found that false news does not spread without believers. The

study analyzed Twitter accounts with high bot followers and identified credulous users based on account characteristics like tweet, friend, and follower counts. Credulous users amplified bot material more than non-credulous users, as confirmed by two statistical tests on account retweets and replies.

The studies mentioned in the text have a few potential weaknesses, including the possibility of bias. While ML and statistical analysis are used to detect fake accounts and misinformation, these methods are susceptible to bias, which may upset the result's accuracy and the dependability of the proposed models. Additionally, the text does not include a comparison with other existing approaches, which limits the assessment of the effectiveness of the proposed methods and their potential impact in the field. A comparison with other approaches would enable a more comprehensive evaluation of the proposed methods.

### 2.10.2 Features Selection Category

To reduce the dimensionality of datasets and identify a subgroup of features that effectively describe the data, FS is utilized [157]. The primary objective of FS is to generate a concise subgroup of features that precisely represent the essential aspects of the entire dataset. As depicted in Figure 2.10, FS algorithms are classified into three main categories: search strategy-based, learning model-based, and manual-based.

1) Manual FS: In two studies, FS was carried out manually to detect Twitter bots [155], [156]. However, a different study [158] proposes using one-class classification to identify bots, which only requires samples of legitimate accounts and doesn't need examples of anomalous behavior. One-class classifiers can distinguish bots from humans and achieve a performance score above 0.89, using a

combination of text, nominal, and numeric data features. It's worth noting that one-class classifiers were only used for numeric data.

2) Search Strategy: FS algorithms use search methods to choose a subset of features, grouped into three types: complete, randomized, and heuristic search. Complete search looks at the entire search space, which may be impractical for high-dimensional datasets, while randomized search explores a smaller subspace with a stopping criterion. To address these issues, GAWA was introduced, integrating Genetic Algorithm and Wrapper Approaches [159]. GAWA uses two wrapper techniques and a modified fitness value in the Genetic algorithm to extract optimal features from Twitter data. The Genetic algorithm pruned 8,243 feature sets to 3,137. Heuristic search algorithms add or remove one feature at a time, making them less computationally expensive than complete search methods. Heuristic search algorithms like a Binary Grey Wolf (BGW), binary moth flame (BMF), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Cuckoo Search (CS), and Genetic Algorithms (GA) have gained popularity due to their global search capability for high-dimensional data. These algorithms can be tailored to specific tasks and prevent early convergence [159]–[162]. By operating as a black box, they avoid local optima while effectively exploring the search space. This approach achieves a balance between exploration and exploitation, as promising search spaces are explored during the exploration phase, and promising local areas are searched during the exploitation phase [163].
3) In relationship with the learning model-based on the domain of statistics and ML, the process of FS is alternatively referred to as

attribute selection, variable subset selection, or variable selection. [164]. FS involves selecting a relevant subset of features for model construction. There are four types of FS methods: wrapper, filter, hybrid, and embedded. The initial feature set can be classified into four groups: features that are irrelevant and produce noise, features that are weakly relevant and redundant, features that are weakly relevant but non-redundant, and strongly relevant features. Wrapper methods use a classifier's performance to select the best collection of discriminative features by reducing the classifier's estimation error [159], [161], [165], [166]. Wrapper methods achieve superior performance and accuracy by incorporating FS into the learning algorithm and assessing feature quality based on performance accuracy or classification error rate. In contrast, filter methods evaluate features before learning tasks using criteria such as information, dependency, consistency, and distance to assess their intrinsic qualities. [166]–[168]. The filter method performs FS independently, making it more efficient and scalable than the wrapper method for high-dimensional datasets. However, it disregards the connection between the selected subset and the algorithm's performance, which is a drawback. Hybrid and ensemble methods for FS can be created by combining two different methods or two methods with the same criteria. This can be achieved by integrating two different methods or two methods with the same criteria [154], [168]. The hybrid method combines the complementary capabilities of both methods and inherits their advantages [169]. One commonly used hybrid method is the grouping of filter and wrapper approaches, as demonstrated in studies [166]. The embedded method is an FS technique that integrates FS into the learning process to guide feature evaluation. In contrast to

the wrapper method, the embedded method does not require running the classifier many times to evaluate each feature subset, making it computationally more efficient. A study that combined binomial linear regression and PCA demonstrated the effectiveness of the embedded technique [151].

## 2.11 Merging features

The merged features of Twitter are the combination of tweet text and metadata features that provide a comprehensive understanding of a tweet, including its content, context, and performance, such as user information, posting time, engagement and reach metrics, tweet length, language, sentiment, and multimedia elements [170]. Two methods were discussed that have merged these features, which were analyzed and categorized based on different previous works. The first method involves merging metadata and tweet text, while the second method extracts features from tweet text and selects a subset of those features. Despite the increasing popularity of these fusion techniques, the literature still depends on a single criterion, which involves assessing features based on specific criteria and selecting the subset of features that performs best.

### 2.11.1 A Fusion of Twitter Metadata and Text Features

Bot detection typically involves merging tweet text features extracted through NLP with Twitter metadata to enable ML techniques to accurately predict malicious bots. While no standardized set of features guarantees good performance, studies introduce specific feature sets ideal for their chosen classifier. Poor feature selection can lead to high computational costs, data overfitting, and a decline in predictor performance. A review of bot detection methods using shallow and deep learning techniques was provided in [171], highlighting the importance of proper feature

selection. To improve bot classification, one study summarized 59 features for building a feature model, including tweet text, tweet date and time, and Twitter account metadata [172]. Another study proposed a deep learning model consisting of three stages for detecting social bots with nearly perfect accuracy (more than 99%) using tweet combined features and tweet user information temporal features [173]. However, social bot identification based on deep learning requires a vast number of tweets and integrating multiple datasets. A third study classified features into four categories based on their attributes and expressed them quantitatively to obtain numerical features mapped to the [0,1] interval using maximum and minimum normalization to solve differences between eigenvalue types and sizes [166]. In the end, additional research put forward a method to identify bots on Twitter using a multilingual strategy and deep learning techniques. The approach involves creating an encoding of the user account's text-based features using deep learning and combining it with metadata to generate a possible input vector. The input vector is then fed into a Dense Network called Bot-DenseNet [174].

### 2.11.2 A Fusion of Extracting and Selecting Features

The combination of NLP and FS methods is widely used to extract tweet text features and reduce dimensionality to improve ML accuracy. In [175], population-based meta-heuristic algorithms, TFIDF feature extraction, and binary grey wolf (BGW) FS are utilized. However, traditional FS methods may cause information loss by discarding low-information features, as discussed in [39]. To address this, [39] introduces fuzzy cross-entropy to preserve information. In [176], deep learning models with automatic FS are used to determine opinion polarity, and a pre-processing step and FS technique are combined to improve data quality and achieve good results using Deep Belief Networks.

## 2.12 Discussion

The use of ML algorithms has become increasingly popular in detecting and predicting various phenomena in social networks. However, selecting the appropriate features for these algorithms is crucial for their success. In this context, various studies have proposed different techniques for FS and FE. For instance, while some studies suggest assigning function scores to each feature to choose the feature set with the top scores, others propose using one-class classification or search-based algorithms. Furthermore, the literature mostly focuses on the highest accuracy and reduces the number of features. Additionally, some studies recommend combining metadata features with tweet text features, as NLP on social network texts can be challenging. After reviewing the previous articles on FE and selection techniques, many conclusions have been drawn.

As shown in [151], using PCA alone for FE may lead to the loss of crucial features. Hence, the study suggests assigning function scores to each feature and selecting the set with the highest scores to prevent this problem. Furthermore, improving ML algorithms that can perform correlation analysis among new feature sets and identify more effective variables in the future may enhance the detection rate.

According to a study conducted in [158], one-class classification can be utilized for better Twitter bot detection by using only legitimate accounts as the reference. However, this approach may be vulnerable to a loophole where cyborg accounts, which exhibit both bot and human behaviors, can be used to form a new group that closely resembles genuine accounts, making detection more difficult.

To select features, search-based algorithms are often used, with GAWA being the best algorithm for accuracy at 92% when combined with NB and genetic algorithms according to [159]. However, GAWA may become trapped in a local

optimum despite parameter adjustments. Randomized search-based algorithms are less computationally complex than complete search-based algorithms.

The literature [160]–[162], [175] mostly focuses on two goals: increasing correctness and reducing the number of selected features. However, when employing a multi-objective FS should also consider computing time, intricacy, steadfastness, and scalability.

The wrapper methods discussed in [159], [161], [165], [166] have drawbacks when they are compared with filter strategies, including higher computational complexity and greater sensitivity to overfitting. Wrapper approaches typically involve multiple dimensions and require lengthy computation periods to achieve convergence, making them impractical for large datasets. An alternative approach is the embedded technique, which combines the advantages of both filter and wrapper methods by selecting features during the construction of the mining algorithm, resulting in reduced computational costs.

To improve the accuracy of machine learning techniques in detecting malicious bots, hybrid methods [152], [166], [171], [174], [177], have been proposed that combine tweet text features extracted using natural language processing (NLP) methods with Twitter metadata. However, these methods face a significant challenge in terms of prediction time.

## 2.13 Twitter Datasets Labeled by Usage

To understand bot behavior in social networks, researchers need datasets that contain both genuine and bot accounts. However, obtaining recent and sufficiently large datasets with diverse bot content is time-consuming and challenging. Creating a well-labeled training dataset is also difficult and slow, as human annotation is prone to error and deception. Additionally, the constantly evolving nature of bots

requires an up-to-date dataset, making it difficult to achieve a consistent strategy of labeling. The absence of a dataset that serves as a reference point is a notable drawback for researchers. The summarized information about Twitter datasets used in previous literatures condensed overview of the Twitter datasets employed in prior studies was presented in Table 1 of Appendix A and serves as a valuable resource for researchers seeking appropriate datasets for future studies. This compilation provides key details that can aid in identifying suitable datasets, making it easier for researchers to select the most relevant data sources for their specific research endeavors.

## 2.14 Exploring the Effectiveness of Bot Detection Techniques

The bot detection accuracy is contingent on the technique used along with FS. ML methods, particularly tree-based approaches, SVM, and Bayes theorem are commonly used by researchers as illustrated in Figure 2.11. The SVM classifier is frequently used but requires a large training set, while RF is simple and accurate but can over fit. Bayes theorem is fast and effective for small datasets, but performance is influenced by labeling accuracy and bot types. The effectiveness of a method cannot be solely determined by performance measures, as training and testing datasets and FS can greatly impact results. The literature describes the classification methods used, as explained in Table 2.5.

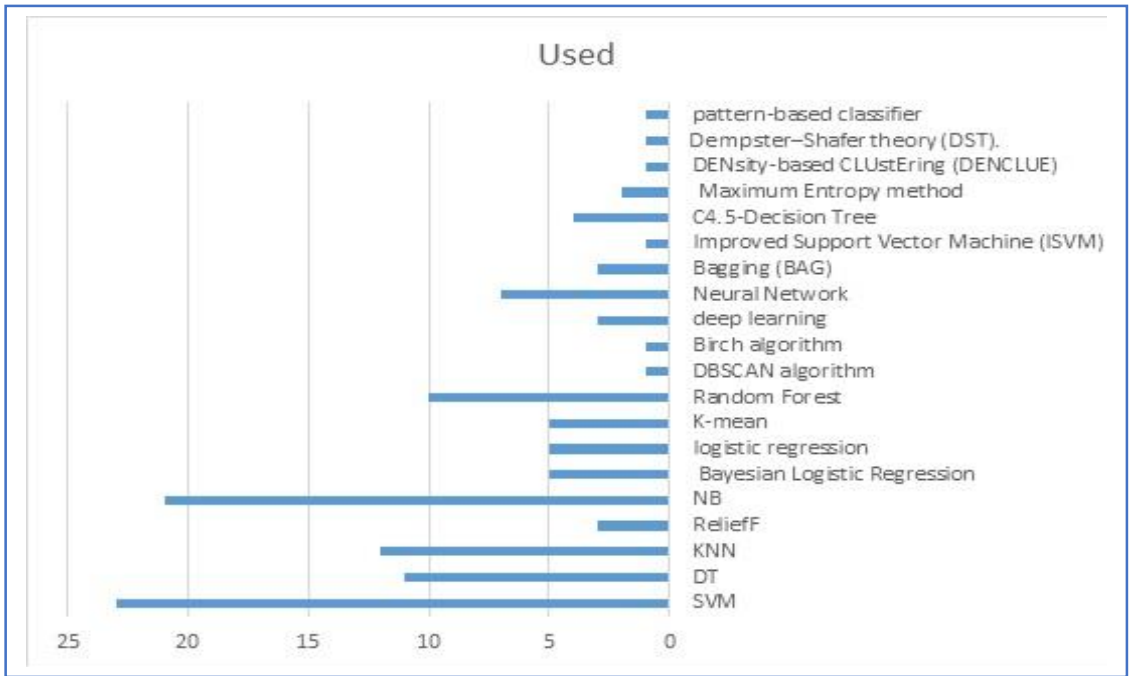


**Figure 2.11: Overview of Classifiers Utilized in the Literature Under Review**

**Table 2.5: Techniques for Categorization Employed in the Literature Under Review**

| Ref | Type of Bots | Approach | Species | Result |
|---|---|---|---|---|
| [178] | Detecting Twitter bots | dbscan and k-mean | Unsupervised | AUC= 97.7. |
| [39] | Spam Twitter Detection | CIFAS | Unsupervised | AUC= 97.11 |
| [179] | Offensive language detection on Twitter | SVM and NB | Supervised | F1= 0.76 |
| [180] | Detecting Twitter bots | LA-MSBD | Supervised | AUC= 0.971 |
| [158] | Detecting Twitter bots | OCSVM | Supervised | Precision= 97.2 |
| [151] | Detecting Twitter Spam | RF, DT, SVM | Supervised | AUC= 85.8 |
| [181] | Detecting Twitter bots | NN model, BiLSTM | Supervised | AUC=0.961 |
| [47] | Detecting Twitter bots | RF | Supervised | AUC= 92.6 |
| [171] | Detecting Twitter bots | RF, DL | Unsupervised | AUC= 99.9 |
| [167] | Detecting Twitter bots | CPB | Supervised | AUC= 98.31 |
| [156] | Detecting Twitter bots | SVM, RF | Supervised | AUC=0.74 |
| [182] | Detecting Cyberbullying | SVM , NB ,RF | Supervised | AUC= 90.84 |
| [183] | Detecting Cyberbullying | CNN | Supervised | AUC= 96.38. |
| [150] | Detecting Twitter bots | ISVM | Supervised | Accuracy= 0.900 |
| [152] | Detect COVID-19 Misinformation on Twitter | NB, kNN, DT, RF SVM. | Supervised | AUC= 86.6 |
| [184] | Credibility Evaluation of Twitter | Mixing Analysis | Unsupervised | F-measure=0.711 |
| [154] | Detecting Twitter bots | RF, NBayes, and SVM | Supervised | AUC= 98%. |

| | | | | |
|---|---|---|---|---|
| [174] | Detecting Twitter bots | DNN | Supervised | F1-Score=0.77 |
| [173] | Detecting Twitter bots | CNN, RNN | Hybrid | AUC= 99%. |
| [176] | opinion polarity | DNN | Hybrid | AUC= 81.7. |
| [185] | Malicious activity | DNN, NLP | Hybrid | AUC= 0.9898 |

## 2.15 Detecting Social Bots: Common Features and Categories

The detection of social bots depends on a set of features that are chosen to distinguish between authentic and bot accounts. Typical features include factors such as timing, text usage, and sentiment. It is important to note that a social bot cannot be classified based on a single feature without taking into account other factors [186]. Table 1 in Appendix B provides a summary of the common features extracted from the full set of features in the reviewed papers, which are used to determine whether an account is human or a bot.

FS is of utmost importance for the accuracy of ML algorithms, particularly in the realm of unsupervised algorithms like clustering, and it holds special significance when it comes to detecting Twitter bots. The process of FS entails carefully picking out a subset of pertinent features from the dataset at hand, to improve the algorithm's overall performance. In the specific context of identifying Twitter bots, the selection of appropriate features can substantially bolster the accuracy of clustering algorithms by effectively distinguishing between authentic user accounts and bot accounts. By strategically choosing the most relevant features, the algorithm becomes capable of capturing the crucial characteristics that set bots apart from genuine users, leading to outcomes that are more precise, trustworthy, and dependable in the detection of bot accounts.

To detect bot accounts on Twitter, four types of features can be used: metadata (e.g., account creation date, follower count), text tweet (e.g., word frequency, sentiment analysis), network (e.g., follower/following ratio, retweet patterns), and

temporal (e.g., tweet frequency, timing). These features provide insights into activity patterns and help identify suspicious behavior. By considering all these features, we can construct a comprehensive representation of a Twitter account for effective bot detection.

## 2.16 Methods of Evaluating Clustering Algorithms

It is crucial to perform a clustering evaluation to assess the accuracy and effectiveness of the produced clusters. There are various metrics used in the proposed system to evaluate clustering models, including homogeneity metric, completeness metric, v-measure metric, adjusted-rand metric, adjusted mutual info metric, silhouette metric, and Fowlkes mallows metric.

1) Homogeneity metric: The assessment measures the degree to which each cluster comprises only data points that are part of a single category or class. A clustering model having a high homogeneity metric suggests that each cluster comprises set points sorted into the same class, whereas a low homogeneity metric indicates that clusters contain mixed data points from different classes.
2) Completeness metric: It measures how well all the data points belonging to the same class are assigned to a single cluster. A clustering model having a high completeness metric indicates that all data points from a single class are placed in the same cluster, while a low completeness metric suggests that data points from the same class are dispersed across multiple clusters.
3) V-measure metric: It is a harmonic mean of the homogeneity metric and completeness metric. A clustering model with a high v_measure metric indicates that the generated clusters are both homogeneous and complete.
4) Adjusted rand metric: It quantifies the similarity between the predicted labels of the clustering model and the true labels. A clustering model having

a high adjusted-rand metric indicates that the predicted labels are highly comparable to the ground truth labels.

5) Adjusted mutual info metric: It is an additional measure to evaluate the similarity between the predicted labels and the actual ground truth labels. This score takes into account the possibility of accidental agreement between the predicted and ground truth labels. A clustering model having a high adjusted mutual info suggests that the predicted labels are comparable to the ground truth labels and not merely due to chance.
6) Silhouette metric: It is the quality of the clusters created by a clustering model by examining the proximity of data points within a cluster to the closeness of data points in other clusters. A high silhouette_score indicates that set points within the Intra-cluster are closer to each other than they are to data points in other clusters.
7) Fowlkes Mallow's metric: It is a similarity measure between the predicted labels and the true labels. It calculates the precision-recall harmonic mean of the clustering model. High Fowlkes mallow metric indicates that the predicted labels are highly comparable to the ground truth labels.

In conclusion, these evaluation metrics are critical for assessing the quality of the clustering models. They help determine the model's accuracy and its ability to generate meaningful clusters from input data. It's essential to use these metrics to select the best clustering model that suits the application's requirements.

## 2.17 Summary

This Chapter explores methods for detecting Twitter bots and highlights important features for accurate identification. It examines existing literature on bot detection based on datasets, features, classifiers, and performance measures. The objective is to develop a new architecture for feature extrapolation to improve ML

applications for bot detection. The chapter provides detailed information on Twitter features that interact with malicious bots classification and identifies benefits, challenges, gaps, and recommendations for bot detection. Solutions are presented to address the identified challenges. Overall, it offers a comprehensive overview of Twitter bot detection, encouraging the application of AI for this purpose.

Furthermore, this chapter provides insight into clustering algorithms and their unique solutions for dealing with the scarcity of addressable data in real-world scenarios. Addressing such scarcity requires significant effort and cost. However, the effectiveness of these algorithms largely depends on the selection of appropriate features as inputs for clustering. This can be elaborated through three points: Relevant features, Feature normalization, FS: Not all features may be equally important for clustering. Using feature selection techniques to identify and select the most relevant ones may help improve algorithm performance and reduce computational costs.

Generally, four key points for selecting influential features for effective clustering are:

1) Dimensionality reduction: Reducing features while preserving information for faster and better clustering.
2) Feature engineering: Creating new features to enhance data representation and clustering performance.
3) Data preprocessing: Improving data quality through cleaning, normalization, and outlier detection for better clustering results.
4) Iterative refinement: Multiplying iterations of FS, engineering, and preprocessing enhance clustering accuracy and quality.

# Chapter Three

# The Proposed System and Methodology

## 3.1 Overview

This chapter focuses on the proposed approach for detecting Twitter malicious bots using unsupervised machine-learning techniques. It depicts the proposed system phases with details about the development of each phase. The adaptation and integration made for different techniques are explained as well.

## 3.2 The Proposed System

The proposed system presented in Figure 3.1 consists of five distinct phases, each designed to ensure efficient implementation:

1) The preprocessing phase: Raw data is prepared and transformed into a format that can be used for analysis or further processing;
2) Features analysis phase: In this phase, the creation of new features and selection of the best ones to enhance clustering algorithms are explained;
3) Experiment Phase: This phase evaluates the capacity of unsupervised cluster algorithms in detecting bots using suggested features and determines the most robust features among them.
4) Development phase: Here, the improvement of three unsupervised algorithms (KNN, Stream k-means, and DBSCAN) is highlighted;
5) The evaluation phase: It measures the proposed approach's performance and identifies the results.

Malicious social media bots, controlled by a botmaster, disseminate harmful content, like illegal link-laden spam. Detecting and thwarting these threats demands advanced methods like unsupervised machine learning clustering to overcome traditional model constraints. Due to bots' human-like mimicry, distinguishing them from genuine accounts is tricky, underscoring the importance of precise feature

extraction and selection in machine learning for bot detection. Some attributes can appear similar in both human accounts and bots.

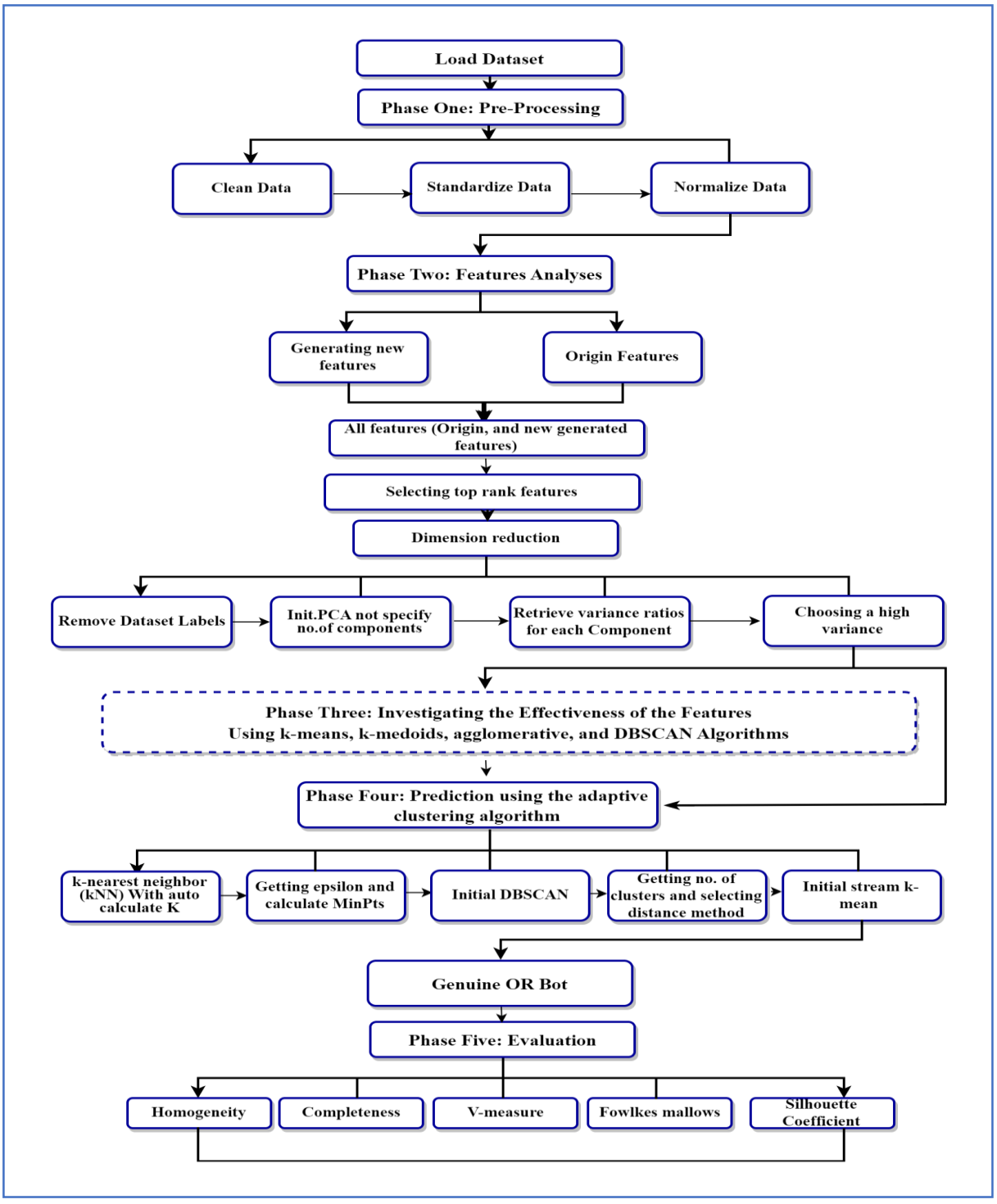



**Figure 3.1: A General Overview of the Proposed System**

The present dissertation proposes a system that is illustrated in Figure 3.1. It operates under the assumption that Twitter data in real-world scenarios are not labeled. This system aims to detect bots using unsupervised models, specifically cluster approaches, that do not require labeled data. The key principle behind these models is to identify similarities between accounts in a given cluster. The effectiveness of the predictions generated by these algorithms is determined by the data's readiness and the identification of crucial features. Thus, the suggested system is divided into two main parts.

The first part concentrates on the preparation of the essential features of Twitter accounts and it comprises three stages. Firstly, the data is prepared. This is followed by the creation of novel features in the second stage. Lastly, the most successful features are pinpointed to aid cluster algorithms in the precise identification of bots. Within this stage, PCA is used to decrease the number of dimensions in the data thus effectively decreasing the amount of computational time necessary for the clustering algorithms.

Before moving to the system's prediction phase, which focuses on distinguishing between human and bot Twitter accounts, the selected features undergo evaluation via three distinct clustering methods: partitioning, hierarchical, and density-based techniques. These tests ensure their compatibility with unsupervised machine learning. In the prediction phase, new integrated unsupervised algorithms (KNN, DBSCAN, and Stream K-means) are employed. The aim of this integration is to capitalize on the unique capabilities of each algorithm and address limitations that a single approach may not overcome.

### 3.2.1 The Preprocessing Phase

Data preprocessing involves getting data ready for a particular purpose, such as ML algorithms by transforming unprocessed data into an understandable format.

The initial stage of this process is crucial as it sets the foundation for the accuracy of the machine learning model. To ensure good data quality, it is essential to assess data completeness, consistency, believability, and interpretability before implementing machine learning models. Some of the essential tasks involved in data preprocessing are as follows:

1) Cleaning of Data: It refers to the process of eliminating duplicate or unrelated information, padding in gaps caused by missing values, and rectifying inconsistencies.
2) Transformation Data: converting data into a numerical or categorical format.
3) Data reduction: simplifying the dataset by removing irrelevant or redundant features.

Algorithm 3.1 begins by importing the necessary data processing tools and defining the input (CAVERLEE Twitter dataset) and output (pre-processed dataset). It removes irrelevant columns such as target and User IDs. Next, the data is cleaned by filling missing values with column means. Finally, it transforms the data for numerical processing, preparing it for further analysis.

| **Algorithm 3.1: Pre-processing the CAVERLEE Twitter Dataset** | |
|---|---|
| | Input: X - CAVERLEE Dataset<br>Output: Pre-Processed Dataset |
| 1 | Import the necessary libraries, tools, and functions required for data processing. |
| 2 | Define the input dataset as the CAVERLEE Twitter dataset (X). |
| 3 | Define the output as the pre-processed dataset. |
| 4 | Load the CAVERLEE dataset into a data structure suitable for analysis (e.g., a data frame or data table). |
| 5 | Remove the target column (categories column) from the X dataset, as it will not be used in the preprocessing. |
| 6 | Remove any columns that contain User IDs, as they are not relevant for the preprocessing. |
| 7 | Clean the dataset by replacing missing or null values with the mean value of each column with missing data. |

| | |
|---|---|
| 8 | a. For each column in dataset X, iterate through all the columns.<br>b. Calculate the mean value of the column if it contains NaN (null) values.<br>c. Replace the NaN values in the column with the mean value.<br>d. Apply the changes in place (modify the original dataset X).<br>e. After this step, the dataset X is cleaned and ready for further processing.<br>Transform the dataset X into a format suitable for numerical processing, ensuring it can be effectively used for subsequent operations. |

**A) Cleaning Data**

It refers to the procedure of recognizing and removing erroneous, incomplete, and imprecise data from datasets, as it can significantly impact the effectiveness of machine learning algorithms. The method used here involves replacing the Empty with the mean value of the relevant column or row, which is the approach taken in the proposed system.

**B) Transformation Data**

Data transformation entails transforming unprocessed data into a uniform structure that is appropriate for analysis and modeling purposes. This is because machine learning algorithms require input data to be consistent in structure and type, which can only be achieved through data transformation. The complexity of data transformation can vary resting on various factors, for instance, the volume, type, and complexity of data, as well as the specific needs of the project. Therefore, it is important to choose appropriate data transformation methods based on these factors.

In the proposed system, two distinct data transformation methods are carefully selected that meet the unique needs of this work. These methods have been selected on the grounds of their ability to transform data into a format that is optimized for machine learning algorithms, ensuring that the transformed data is effectively utilized in the analysis. By transforming data using appropriate techniques, it can be ensured that it is in a format that can be effectively analyzed by machine learning

algorithms. This, in turn, enables us to derive valuable insights and outcomes from the data, ultimately leading to better outcomes and results for this system.

1) To standardize data, one needs to convert it to a uniform scale or interval by deducting the average and dividing it by the standard deviation. This is important in statistical and machine-learning applications to ensure accuracy and efficiency by comparing features that have different units or scales. Standardization also facilitates data visualization, aids in identifying outliers as they will have values that are significantly different from the standardized mean, and ensures consistency and comparability across different datasets. The equation for StandardScaler is given as [187]:

$$\mathbf{z = \frac{x - u}{s}} \tag{3.1}$$

where

x feature value

u mean feature values

s Standard Deviation (ST)

z feature standardized value also called the z-score

Using the StandardScaler ensures that each characteristic of the data has an average of 0 and an ST of 1. The mean and standard deviation are calculated for each feature separately from the training data. Once calculated, the same mean and standard deviation are used to transform the test data, ensuring that the scaling is consistent across the training and test datasets.

2) Normalization: Normalization is a technique used in data processing to make data consistent across all fields and records. Its primary objective

is to establish connections between entry data, which improves data quality. The technique involves scaling the data into a smaller range, such as -1 to 1, making it easier to represent and work with. Equation 3.2 shows the normalization function [187]:

$$\mathbf{x\_norm} = \frac{\mathbf{x}}{||\mathbf{x}||} \tag{3.2}$$

where,

x is the vector to be normalized

||x|| is the L2 norm of an x vector, which is described as the radical of the squared sum of the vector's elements.

## 3.3 Principle Component Analysis (PCA)

PCA is a statistical technique that helps simplify high-dimensional data by identifying the most important features. This technique offers three main benefits: 1) it reduces the number of features, 2) it improves algorithm training time, and 3) it enhances the accuracy of model predictions. Additionally, this technique simplifies data visualization, which can facilitate a better understanding of the data and model.

The essential objective of PCA is to capture the most significant amount of variability present among the features within a given dataset. However, it is crucial to select an appropriate number of Principal Components to prevent the loss of critical information. The number of components to retain can be influenced by factors such as storage capacity, training time, and performance.

Algorithm 3.2 leverages PCA to reduce the dimensionality of the input dataset while retaining a significant portion of the original data's variance. This dimensionality reduction is useful for simplifying complex datasets and improving computational efficiency in subsequent analyses. The system selects Principal Components with the highest percentage of variance and disregards others.

| Algorithm 3.2: Variance-Based Feature Selection | |
|---|---|
| | **Input: Dataset from A pre-processed Algorithm.**<br>**Output: A reduced dataset with fewer features.** |
| 1 | Import the PCA class for dimensionality reduction. |
| 2 | Initialize PCA without specifying the number of components initially. |
| 3 | Retrieve explained variance ratios for each PCA component. |
| 4 | Set a target of retaining 80% of dataset information and use a counter (D). |
| 5 | Iterate through variances, starting with the first component. Stop and return D when cumulative explained variance surpasses 80%, otherwise, continue adding variances. |
| 6 | Create a new PCA instance with the chosen components in the previous step. |
| 7 | Transform the original dataset using this new PCA instance. |
| 8 | Reconstruct the reduced dataset, renaming columns as Component-1 and 'Component-2'. |
| 9 | The final reduced dataset is now ready for analysis and modeling. |

Figure 3.2 of the proposed system shows variance percentages for each of the five features [0.67120865, 0.1912806, 0.10383127, 0.03237566, 0.00130383]. The first feature accounts for 70% of the information, and the second feature represents 20%. By combining these two features, they cover 90% of the total information, which is considered good. Therefore, the proposed system will only rely on the first and second components because they effectively reduced the data dimensions to two parts.

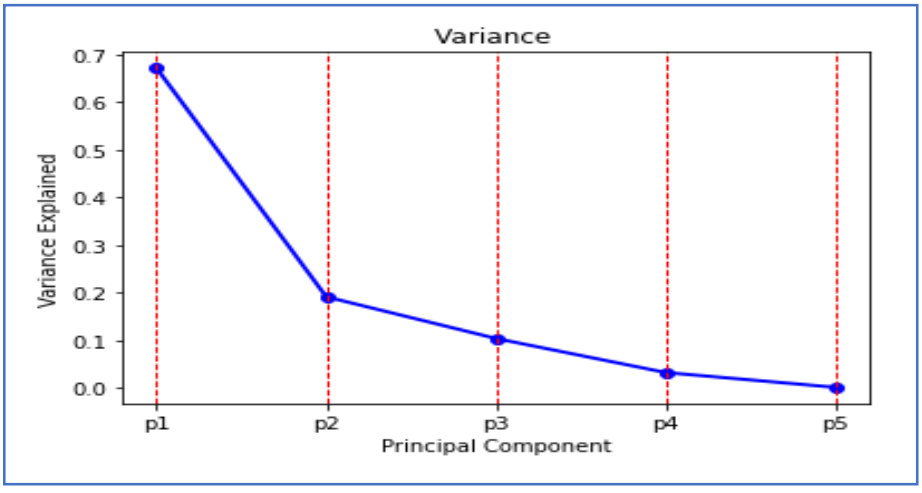



**Figure 3.2: The Principal Components Variance Change**

## 3.4 Features Analysis Phase

Figure 3.1 illustrates that the Feature Analysis stage comprises two layers: the Feature Generation and the Feature Selection layers. The tweet metadata features are produced in the Feature Generation layer, and the most highly-ranked features are chosen from the entire set of features in the Feature Selection layer. These specifics are expounded upon in Sections 3.5 and 3.6.

### 3.4.1 Feature Generation

This involves selecting and extracting the most relevant and meaningful information from the raw data, so it can be easily understood and used by the mode. One effective strategy for managing clusters is to extract essential features. This approach can help to accurately identify clusters, gain a deeper perception of the data, and reduce the data dimensionality.

To achieve this, the proposed system generates a variety of Twitter features that fall into one of four categories: User Property Features (UPF), User Neighborhood Features (UCF), User Content Features (UCF), and User Hybrid Features. For clarity, Table 3.1 provides a comprehensive outline of the specific features generated by this system, which can be used in various machine-learning applications.

**Table 3.1: Categories of Twitter Features Generated by the Proposed System**

| Feature Category | Description |
|---|---|
| **UPF** | Features extracted from the user profile information such as , user name, account age, biography, location, and verification . |
| **UNF** | FE from the user's connections such as the follower's number, friends number, and mutual friends. |
| **UCF** | FE from the text posted by the user such as tweets number, retweets, likes, replies, and hashtags used. |

| | |
|---|---|
| **User Hybrid Features** | Features that combine information from multiple categories such as the average number of likes per tweet or the ratio of followers to friends. |

### A) User Property Features (UPF)

This section provides descriptive information about a user and his/her account, including age and Twitter profile information:

1) Age_month: The account's age is determined by a monthly value that is calculated using Equation 3.3 [188] based on the duration between the time the account was retrieved and its creation date.

$$\mathbf{age_{month}} = (\mathbf{CreatedYearAt} - \mathbf{CollectedYearAt} * \mathbf{12}) + (\mathbf{days\ per\ month}/\mathbf{30}) \quad \textbf{(3.3)}$$

where 30, is the average duration of a month. When calculating age in months, it is typically divided by the number of days by 30 (average days number in a month).

The age of a Twitter account is a crucial factor in determining its credibility and identifying malicious bots on the platform. Generally, older accounts are considered more trustworthy than newly-created ones. This metric is utilized in the proposed system to derive additional features like "CV_Following," "FollowingToAgeRate," "Avg_tweets," and "Bfr_afr."

2) Age_days: The age of a Twitter account is determined based on the duration between the time it was created and the date at hand. Typically, this time frame is indicated concerning the number of days that have elapsed since the account's creation. To calculate the daily age of a Twitter account, Equation 3.4 is employed, which takes into account the account's creation date and the current date. This calculation results in a numerical

value that represents the daily age of the account, which is then used to determine the account's credibility and authenticity.

$$\mathbf{age\_days} = (\mathbf{CreatedYearAt} - \mathbf{CollectedYearAt}) * \mathbf{365} \quad \textbf{(3.4)}$$

where 365 is the number of days in a year on average, considering the leap years as well.

3) Pro_Info: The profile information feature is used to evaluate the credibility of a Twitter account and is assigned a numerical value based on the information present in the account's profile. Typically, four types of information are usually provided by humans when creating a Twitter account, which includes a bio, name, location, and account verification status (Equation 3.5). In the proposed approach, a value of 1 is assigned to each of these pieces of information if they are present in the account's profile, whereas a value of 0 is assigned if any of these pieces of information are missing.

$$\mathbf{Pro\_Info} = (\mathbf{bio} + \mathbf{name} + \mathbf{location} + \mathbf{verification}) \quad \textbf{(3.5)}$$

Where bio is a biography which represents a personal description of a profile. It appears that a small number of legitimate accounts leave this feature, whereas a large number of bot accounts leave this feature. Additionally, verification A verified account, according to Twitter, refers to any account that is of relevance to the general public and has been authenticated by the organization, to receive the blue badge, your account must be authentic, notable, and active[189], [190].

**B) User Neighborhood Features (UNF)**

The most significant indicator of a Twitter account's social influence is typically reflected in its friendship information, which includes the number of

followers and following accounts. In this section, five distinct features have been prposed to evaluate this aspect of a Twitter account:

1) MaxMinfollowing: The MaxMinfollowing feature is employed to calculate the range of numbers of following accounts for a Twitter account based on its age. It has been observed that in human accounts, this difference tends to remain relatively stable over time. However, in the case of bot accounts, this difference can be significant and unstable. Thus, the MaxMinfollowing feature can be useful in distinguishing between human and bot accounts as shown in Equation 3.6.

$$\mathbf{MaxMinfollowing} = [\mathbf{MAX}(\mathbf{UsersFollowings}) - \mathbf{MIN}(\mathbf{UsersFollowings})] \quad \textbf{(3.6)}$$

2) CV_Following: One of the key features used to assess the credibility of a Twitter account is the CV_Following metric. This feature is calculated using a coefficient variation (CV) to calculate the monthly variability in the number of accounts that user-following. A Twitter account with a high CV value is considered to be unstable and may be classified as a bot. The coefficient of variation is used in this feature because the average number of tweets per user may vary significantly based on the account's age. Equations 3.7, 3.8, and 3.9 are used to calculate this feature by taking into account the differences in tweet averages across accounts.

$$\mathbf{Stander\ Deviation}(\mathbf{Sd}) = \sum_{i=1}^{n} \frac{(\mathbf{Xi}-\mathbf{average})\text{^}2}{\mathbf{AccountsAgeMonths}} \quad \textbf{(3.7)}$$

$$\mathbf{Coefficient\ variance} = \frac{\mathbf{stander\ deviation}(\mathbf{Sd})}{\mathbf{average}} \quad \textbf{(3.8)}$$

$$\mathbf{Average} = \frac{\mathbf{SumMonthFollowings}}{\mathbf{AccountsAgeMonths}} \quad \textbf{(3.9)}$$

where n is the number of records, i is the first record, AccountsAgeMonths represents the age of account by months, $X_i$ is the number-of-following for each month, and SumMonthFollowings is the number of following per month.

3) The objective of deriving the "following_vs_followers_ratio" feature is to indicate the equilibrium between the count of accounts that a user tracks and the number of accounts that follow that user. A well-balanced ratio is considered more typical for human accounts, whereas bots may exhibit an imbalanced ratio (see Equation 3.10). Generally, a stable ratio for human accounts is expected to fall within the range of 0.75 to 1.25 [191].

$$\textbf{FRatio} = \frac{\textbf{Number of Followers}}{\textbf{Number of Followings}} \quad \textbf{(3.10)}$$

4) FollowerToAgeRate: The FollowerToAgeRate feature is computed by calculating the average number of followers gained by a Twitter account per month. This feature is useful in evaluating the credibility of a Twitter account since it indicates the rate at which the account is gaining followers. It has been observed that the follower ratio for human accounts tends to be higher compared to bot accounts. Spammers frequently have imbalanced following/follower ratios, as they follow large numbers of other users but they have comparatively few followers. A high follower-to-age ratio could potentially indicate a bot, especially if the account has gained a large number of followers in a short period of time. This is because bot accounts may engage in tactics such as mass following and spamming to increase their follower count quickly, whereas human accounts tend to gain followers at a more gradual pace. However, this is not always the case as popular accounts or those of public figures can also have high follower-to-age ratios.. This feature is computed using Equation 3.11.

$$\mathbf{FollowerToAgeRate} = \frac{\mathbf{NumberofFollowers}}{\mathbf{AccountsAgeMonths}} \quad \textbf{(3.11)}$$

5) FollowingToAgeRate: The FollowingToAgeRate feature is used to calculate the average number of accounts-following following a Twitter account per month. This feature is important in evaluating the credibility of a Twitter account since it indicates the rate at which the account is following other accounts. It has been observed that the following ratio tends to be lower for bot accounts compared to human accounts. This is because bot accounts often follow a large number of accounts in a short period to appear more active and gain more followers. In contrast, human accounts tend to follow other accounts at a more moderate pace. Equation 3.12 is established to calculate this feature.

$$\mathbf{FollowingsToAgeRate} = \frac{\mathbf{NumberofFollowings}}{\mathbf{AccountsAgeMonths}} \quad \textbf{(3.12)}$$

## C) User Content Features (UCF)

Descriptive information about tweets posted information, including the number of tweets, retweets, length of tweets, etc., four features created in this section.

1) The Avg_tweets feature calculates the average tweets number posted by a Twitter account per month. This feature is important in determining the credibility of a Twitter account since it indicates the account's activity level. It has been observed that the ratio of tweets from bot accounts to tweets from human accounts is substantially larger. To compute this feature, Equation 3.13 is used:

$$\mathbf{Avg_{tweets}} = \frac{\mathbf{Numbers\_ofTweets}}{\mathbf{AccountsAgeMonths}} \quad \textbf{(3.13)}$$

2) bfr_afr: The bfr_afr feature compares the length of the original tweet with the length of the tweet after removing symbols such as hashtags, HTTP links, and other special characters. This feature is important in determining the credibility of a Twitter account since it indicates the linguistic complexity of the tweets posted by the account (see Equation 3.14). It has been observed that the alphabetic content in bot accounts is shorter compared to human accounts. This may be because bots tend to post pre-written or automatically generated content that is less sophisticated than the content generated by humans.

$$\mathbf{bfr\_afr} = \frac{\mathbf{lenth\ of\ original\ tweet - length\ of\ tweet\ after\ cleaning}}{\mathbf{MonthsNumbers}} \quad \mathbf{(3.14)}$$

3) MaxOfMonth: MaxOfMonth is a metric that measures the maximum number of tweets posted in a month as shown in Equation 3.15. This metric can be useful for analyzing trends and identifying peak activity periods on social media platforms. According to observations, the tweets average posted by bots was higher than the tweets average posted by human users. This suggests that bots may be responsible for a significant portion of the activity on social media platforms.

$$\mathbf{MaxOfMonth} = \mathbf{MAX}(\mathbf{Numbers\ Monthly\ Tweets}) \quad \mathbf{(3.15)}$$

4) MaxOfDays: The maximum number of tweets in a day is calculated based on Equation 3.16. It has been observed that the average number of human tweets was lower than the number of bots tweets.

$$\mathbf{MaxOfDays} = \mathbf{MAX}(\mathbf{Number\ of\ daily\ tweets}) \quad \mathbf{(3.16)}$$

**D) User Hybrid Features**

This area combines features from a user's property, user's content, and user neighborhood, one feature created in this section.

1) By blending the profile information represented by the feature Pro with the percentage of followers of the account, this feature is generated using Equation 3.17.

$$\textbf{ProFollow Ratio} = \textbf{Ln}\left(\left(\frac{\textbf{FollowersNumbers}}{\textbf{FollowingsNumbers}}\right) + \textbf{Pro}\right) * \textbf{100} \qquad \textbf{(3.17)}$$

This feature discusses a method for more accurately distinguishing between Twitter bots and legitimate accounts by incorporating additional information from the account's profile. This information includes the account's bio, name, location, and whether or not it is verified. By combining this profile information with the percentage of followers, the method aims to provide a more accurate indicator of the account's credibility.

The approach of user profile information to differentiate between bots and legitimate accounts is a commonly used technique in social media analysis. Bots often have generic or vague profile information, whereas legitimate accounts typically have more detailed and personalized information. Incorporating this profile information into the analysis can help to identify accounts that are likely to be bots. Adding the percentage of followers as an indicator of the accreditation follower rate can also help improve the accuracy of the analysis.Legitimate accounts typically have a higher proportion of genuine and engaged followers. By combining this information with the profile information, the analysis can provide a more comprehensive understanding of the account's credibility.

Table 3.2 presents a comprehensive summary of all the features that have been generated for the proposed system. The table provides a detailed outline of each distinct feature, along with its name, description, and category. By analyzing the information presented in Table 3.2, one can gain a comprehensive understanding of the various capabilities and functionalities of the system.

**Table 3.2 List of Unique Features of the Proposed System**

| Feature Name | Description | Category | Range of Values befor normalizatio n |
|---|---|---|---|
| **age_month** | The account's age is determined by a monthly value. This metric is utilized to derive additional features like "CV_Following," "FollowingToAgeRate," "Avg_tweets," and "Bfr_afr." | UPF | 0-43 |
| **age_days** | The age of a Twitter account is determined based on the duration between the time it was created and the current date. The age of the account is typically indicated by the number of days that have passed since its inception. | UPF | 0-1330 |
| **Pro_Info** | The profile information feature is used to evaluate the credibility of a Twitter account and is assigned a numerical value based on the information present in the account's profile. Typically, four types of information are usually provided by humans when creating a Twitter account, which includes a bio, name, location, and account verification status. | UPF | 0-4 |
| **MaxMinfollowing** | The MaxMinfollowing feature is employed to calculate the range of numbers of following accounts for a Twitter account based on its age. This feature can be useful in distinguishing between human and bot accounts. | UNF | 0-28171 |
| **CV_Following** | This feature is calculated using the coefficient of variation (CV) to measure the monthly variability in the number of accounts that the user is following. A Twitter account with a high CV value is considered to be unstable and may be classified as a bot. The coefficient of variation is used in this feature because the average number of tweets per user may vary significantly based on the account's age. | UNF | 0-377.94 |
| **FRatio** | The following_vs_followers_ratio feature is used to indicate the equilibrium between the count of accounts that a user tracks and the number of accounts that follow that user. A well-balanced ratio is typically considered more | UNF | 0-1630 |

| | | | |
|---|---|---|---|
| | typical of human accounts, whereas bots may exhibit an imbalanced ratio. | | |
| **FollowerToAgeRate** | The FollowerToAgeRate feature is computed by calculating the average number of followers gained by a Twitter account per month. This feature is useful in evaluating the credibility of a Twitter account since it indicates the rate at which the account is gaining followers. | UNF | 0-239408 |
| **FollowingToAgeRate** | The FollowingToAgeRate feature is used to calculate the average number of accounts-following following a Twitter account per month. This feature is important in evaluating the credibility of a Twitter account since it indicates the rate at which the account is following other accounts. | UNF | 0-13594 |
| **Avg_tweets** | This feature calculates the average tweets posted by a Twitter account per month. This feature is important in determining the credibility of a Twitter account since it indicates the account's activity level. | UCF | 0-84931 |
| **bfr_afr** | The bfr_afr feature compares the length of the original tweet with the length of the tweet after removing symbols such as hashtags, HTTP links, and other special characters. This feature is important in determining the credibility of a Twitter account since it indicates the linguistic complexity of the tweets posted by the account. | UCF | 0-84931 |
| **MaxOfMonth** | MaxOfMonth is a metric that measures the maximum number of tweets posted in a month. This metric can be useful for analyzing trends and identifying peak activity periods on social media platforms. | UCF | 1-381 |
| **MaxOfDays** | The maximum number of tweets in a day. It's been observed that the average number of human tweets was lower than the number of bot tweets. | UCF | 1-221 |
| **ProFollowRatio** | This feature combines the profile information represented by the feature Pro_Info with the percentage of followers | Hybrid | 4.61-16.15 |

### 3.4.2 Feature Selection

To achieve success in machine learning, several factors need to be considered. The most crucial and primary factor is the structure and clarity of the example data. While additional features may increase the ability to discriminate, experience has shown that this is not always the case. If the data contains too much repetitive or irrelevant information or is inaccurate and noisy, it can hinder the learning process during the training phase. To address this problem, a technique referred to as feature subset selection can be employed to recognize and eliminate unnecessary and

duplicated data. This results in less dimensional data, making it easier for learning algorithms to operate at a faster rate. The suggested system proposes three fundamentals for determining the optimal approach to feature selection, which is:

1) Initial point: The choice of the initial point in the feature subset space from which to begin the search is an important consideration, as it can significantly affect the direction of the search. One approach is, to begin with, no features and then gradually add them, allowing the search to progress forward through the search space. Conversely, the search can start with all available features and then gradually eliminate them, moving backward through the search space. This approach can be useful in situations where there are many irrelevant or redundant features, as it can lead to a more efficient search process. Ultimately, the optimal approach for selecting a starting point in the feature subset space will depend on the specific data and problem being addressed.
2) Correlation analysis is a valuable technique that can gauge the intensity of the linear association between two variables. It has applications in machine learning and data science for selecting relevant features, but it is essential to be careful in understanding the findings and exploring the cause-and-effect association between the variables. The Pearson correlation is used here.
3) The most important distinguishing characteristic of feature selection algorithms in machine learning is their approach to evaluating subsets of features. While many algorithms aim to eliminate undesirable features, there is a risk of discarding potentially valuable features in this process. Therefore, the preferred approach involves assigning weights to each feature based on its importance, allowing for a more nuanced and accurate evaluation. These weights are then used by the learning algorithm to pick the most significant

features for the prediction process. This approach is implemented in the proposed system, wherein newly introduced features are combined with existing ones, and the most top-rated features are chosen.

In pursuance of obtaining the essential features that will support detecting bots, the features previously in the database were gathered and named the "original features" alongside the proposed features. Table 3.3 provides detailed information on the features included in the proposed system, along with the groups they represent. The table highlights the specific features that are relevant to each group, allowing for easy comparison and analysis of the different groups.

Under the authority of the Correlation Attribute Eval (CAE) technique [192], it will be ranking these features and choose only positive rank features into account.

**Table 3.3: Feature Details and Associated Groups**

| Id | Taxonomy | Feature name | description |
|---|---|---|---|
| 1 | User Property | LengthOfScreenName | Original features |
| 2 | User Property | LengthOfDescriptionInUserPro | Original features |
| 3 | User Property | CreatedAt | Original features |
| 4 | User Property | CollectedAt | Original features |
| 5 | User Neighborhood | NumerOfFollowings | Original features |
| 6 | User Neighborhood | NumberOfFollowers | Original features |
| 7 | User Content | NumberOfTweets | Original features |
| 8 | User Neighborhood | CV_Following | Proposed Features |
| 9 | User Neighborhood | FollowertoFollowingRatio | Proposed Features |
| 10 | User Hybrid | ProFollow Ratio | Proposed Features |
| 11 | User Hybrid | FollowerToAgeRate | Proposed Features |
| 12 | User Hybrid | FollowingToAgeRate | Proposed Features |
| 13 | User Content | Avg_tweets | Proposed Features |
| 14 | User Content | bfr-afr | Proposed Features |
| 15 | User Content | MaxOfMonth | Proposed Features |
| 16 | User Content | MaxOfDays | Proposed Features |
| 17 | User Property | age_month | Proposed Features |
| 18 | User Property | age_days | Proposed Features |
| 19 | User Property | Max-Min | Proposed Features |
| 20 | User Property | Pro | Proposed Features |

### 3.4.3 Investigating the Effectiveness of the Features

Feature selection is a common approach in supervised learning where class labels are provided, but it can be challenging for unsupervised learning or clustering tasks where the best set of relevant features is not always clear. The objective of picking the most significant attributes for clustering is to identify the most useful relevant features that aid in identifying natural clusters from the data based on the chosen criteria.

This investigation aims to test the ability of unsupervised cluster algorithms to detect bots using the proposed features and identify the strongest features among them. To achieve this, three types of cluster algorithms (partition, hierarchy, and density) were implemented which are represented by four algorithms (k-means, k-medoids, agglomerative, and DBSCAN). The model was evaluated using the original features and then the evaluation was repeated with the proposed features. The results were compared to determine the most accurate algorithm. In addition, we only tested the five highest-ranked features out of the 11 selected properties, and the results were promising, especially in terms of implementation time. To further reduce the dimensionality of the feature space, PCA was used to reduce the size of these five features, resulting in only two dimensions. This greatly improved the speed and precision of the clustering technique in detecting the bot. The data processing, feature extraction, and selection phases are now complete.

## 3.5 Adapting the Clustering Algorithms Phase

Clustering algorithms are a powerful tool for identifying Twitter bots, as they can detect bot-like behavior even in unlabeled data. However, there are limitations and weaknesses in their use, such as feature extraction, scalability, and

interpretability. Therefore, it is recommended to use clustering algorithms in conjunction with other algorithms to improve the accuracy of bot detection.

The proposed system combines KNN, DBSCAN, and stream K-means clustering algorithms to enhance the clustering process. By leveraging the strengths of each algorithm, the system addresses its weaknesses. For example, one algorithm may be better suited to handling outliers, while another may be more adept at detecting subtle patterns. This approach results in more accurate and robust clustering, as each algorithm complements the others' strengths. The integration of all three algorithms produces a comprehensive and dependable clustering that is critical for data analysis and machine learning applications.

### 3.5.1 Choosing Appropriate Clustering Algorithms

Choosing an appropriate clustering algorithm relies on multiple factors, for instance, the nature and data magnitude, the analysis objectives, speed and efficiency, cluster size and number, and cluster shape. Overall, it is important to consider these factors concerning specific data and research questions when choosing a clustering algorithm. It may be helpful to perform an experiment with different algorithms and compare the results to determine which one best meets the research needs. The suggested system combines the strengths of three cluster algorithms - KNN, DBSCAN, and stream K-mean - to mitigate their limitations. This integration offers several advantages and notable features, which are as follows:

1) Simple and easy to modify: The approach is simple and can be easily modified to suit the needs of different datasets and research questions.
2) Many input parameters: Using a few input parameters can make the approach easy to be used and applied.

3) Time complexity: The approach is computationally efficient, making it suitable for large datasets and real-time clustering.
4) Memory space efficiency: The approach is memory space efficient, allowing for the clustering of large datasets without requiring excessive memory resources.
5) Sensitivity to outliers: The combination of KNN and DBSCAN helps to improve sensitivity to outliers, which can be a weakness of K-means clustering.
6) Detecting arbitrary shapes: The combination of DBSCAN and stream K-mean helps to detect arbitrary shapes, which can be a weakness of KNN and K-means clustering.

By combining KNN, DBSCAN, and stream K-mean, this integrated approach provides a more robust and accurate clustering solution than any individual algorithm alone. This approach is particularly useful for datasets with complex shapes, and outliers. It allows for the efficient and effective identification of meaningful clusters that can be used to guide further analysis and decision-making.

### 3.5.2 Improving the KNN Algorithm

The suggested approach involves utilizing the KNN to ascertain the appropriate value of the epsilon parameter for the DBSCAN. The objective is to identify the "elbow" in the plot that corresponds to the ideal epsilon value. The KNN is employed to measure the distance between each point and its kNN. Outcome distances are sorted in ascending order and used to generate a graph that depicts the distance versus the point index. The "knee" of the plot corresponds to the point where the slope of the curve changes, indicating a transition from a high-density region to a low-density region. The distance at the "knee" is then used as the value of epsilon for the DBSCAN approach. This approach allows the automatic selection of the

epsilon parameter without requiring manual tuning, making it a useful tool for clustering high-dimensional datasets.

It is critical to mention that this technique may not always yield the best results, and other methods may be more effective for selecting the value of epsilon for DBSCAN, depending on the particular dataset and problem. Additionally, the choice of k (number of neighbors) in the KNN algorithm can also impact the performance of the method, to address this, the proposed system has improved the KNN algorithm by automatically calculating the optimal value of k using Equation 3.18. It was constructed based on the characteristics of the data, and its validity was confirmed by testing it with the same dataset.

$$\textbf{Number Of Neighbors}(\mathbf{K}) = \frac{\left(\sqrt[2]{\textbf{DatasetLengh}}(\boldsymbol{N})\right)}{\textbf{Getgories Type}(\mathbf{GT})} \qquad \textbf{(3.18)}$$

where N data points number and GT represented the three expected categories of labels (Human Account, Bots account, and unknown account). This Equation is based on empirical observations and has been shown to work well in practice.

By automatically calculating the value of k, the proposed system eliminates the need for manual tuning of this parameter, making it more automated and efficient. Additionally, this approach ensures that the optimal value of k is chosen for each dataset, improving the accuracy and effectiveness of the clustering process.

Algorithm 3.3 takes the dataset from the output Variance-Based Features Selection Algorithm, calculates the optimal epsilon value using a knee technique, and returns the result. The optimal epsilon is an essential parameter for the DBSCAN clustering algorithm, determining the maximum distance between data points for them to be considered part of the same cluster.

The find_knee() function computes the pairwise distances between all points using the pairwise_distances() function and then loops over different values of k. For each value of k, it computes the average distance to the k nearest neighbors for each point in X and then computes the inside cluster squares sum as the average sum of the average squared distances. It stores intra-cluster squares sum for each k in a list "wcss". After computing the sum of the intra-cluster squares for each of k, the find_knee() function plots the curve and finds the knee. Finally, the “knee” of the curve is computed to find the optimal value of epsilon.

| **Algorithm 3.3: Adaptive Unsupervised k-Nearest Neighbor** | |
|---|---|
| | **Input: Dataset from output Variance-Based Features Selection Algorithm**<br>**Output: Optimal Epsilon value for DBSCAN** |
| 1 | Define a function "pairwise_distances" to calculate the pairwise distances between all For each point i_ in X |
| 2 | Determine k-value:<br>Calculate the number of neighbors (k) using the formula: k = sqrt(N) / Type of target variable, where N is the number of data points in the dataset. |
| 3 | Initialize the Nearest Neighbors algorithm with the proposed k from step 2 as the number of neighbors and use the KD-tree algorithm for efficient neighbor searches. |
| 4 | Find the k-neighbors for each data point in the dataset and store the distances to neighbors in neigh_dist and their indices in neigh_ind. |
| 5 | Sort the neighbor distances in ascending order. |
| 6 | Extract the distance of the k-th neighbor and store it in the k_dist list. |
| 7 | Plot the values in k_dist list to visualize the knee point on the plot. |
| 8 | Add a horizontal dashed line at a chosen threshold (e.g., 0.04) to identify the optimal epsilon on the plot. |
| 9 | Return the knee point value as the optimal epsilon for DBSCAN clustering. |

### 3.5.3 Improving the DBSCAN Algorithm

The DBSCAN algorithm's strengths can be summarized in its ability in detecting clusters of arbitrary shape and handling noisy data effectively, and this algorithm does not require a pre-specified number of clusters. Suitable parameter values can be determined through various methods, but they are computationally expensive. In the proposed system, the modified KNN algorithm was used to address the epsilon coefficient problem, while Equation 3.19 was developed to calculate the

Minimum samples (“MinPts”) coefficient, resulting in promising results in terms of algorithm accuracy and time complexity.

Usually, having a MinPts value of 1 doesn't make sense because every point would form its cluster. If we use MinPts=2 and the single link measure, the result would be similar to hierarchical clustering with the dendrogram clipped at a certain height. However, larger MinPts values tend to work better for datasets with noise and produce more significant clusters. The proposed system aims to determine the appropriate MinPts value by considering two factors: the dimensionality of the data (in our case, two dimensions), and the types of Twitter accounts we want to classify (bot, human, or suspicious). Therefore, we have developed Equation 3.19 which takes both factors into account.

$$\mathbf{MinPts} = (\mathbf{2} * \boldsymbol{D}) * \boldsymbol{TTA} \qquad \textbf{(3.19)}$$

The variable "D" in the equation stands for the number of dimensions in the dataset, and it has been multiplied by 2 for two reasons. Firstly, it ensures that the resulting values are as large as possible. Secondly, in case of an error, while loading the data where the dimension value becomes zero, the system will reject it. The second variable in the equation, TTA, denotes the type of Twitter account.

Algorithm 3.4 "The Adaptive DBSCAN Algorithm" shows that the " get number of cluster " function requires two inputs: X, which is the dataset, and I, the number of dimensions obtained from "Variance-Based Feature Selection algorithm ". This function uses the DBSCAN algorithm with three inputs: dataset X, the neighborhood radius eps obtained from "Adaptive Unsupervised k-Nearest Neighbor algorithm" , and the smallest sample number required to form a dense region min_samples obtained from Equation 3.19. The "get number of cluster " function is used to return the number of clusters found by the algorithm.

| Algorithm 3.4: Adaptive DBSCAN Algorithm | |
|---|---|
| | **Input: X - Dataset from Variance-Based Features Selection Algorithm**<br>**Output: Number of Clusters**<br>**Function:** get number of cluster(self, Caverlee dataset, The number of dimensions): |
| 1 | epsilon -- the radius of the neighborhood around each sample from Adaptive Unsupervised k-Nearest Neighbor |
| 2 | min_samples (MinPts)-- MinPts=(2*number of dimension)* Type of target variable |
| 3 | Implementation of the DBSCAN clustering algorithm.<br>clusters = DBSCAN(eps=epsilon, min_samples=MinPts).fit(X) |
| 4 | Getting cluster labels and checking unique clusters: clusters.labels_ |
| 5 | Assign -1 value as noisy points that could not be assigned to any cluster |
| 6 | set(clusters.labels_) |
| 7 | Counting the numbers of clusters: Counter(clusters.labels_) |
| 8 | Mapping origin class with predict label of DBSCAN algorithm |
| 9 | Checking the accuracy of DBSCAN |
| 10 | Counting the number of DBSCAN labels to set it as a number of clusters(K) to Stream K-mean algorithm |
| 11 | Exclude noise points (-1 value) as outliers, and could not be assigned to any cluster |
| 12 | Returns numbers of cluster |

The k-mean algorithm faced two main issues which were addressed by modifying the DBSCAN algorithm. This modified approach can automatically determine the number of clusters (k) and select the appropriate distance equation to determine the centroid. To identify the clusters and outliers, the Canberra distance function was chosen based on a statistical analysis of results obtained from applying the adapted DBSCAN algorithm. Using this clustering technique, two clusters were identified with 20286 points for the largest cluster (0) and 13819 points for the smallest cluster (1), while the number of noise points was 371. It is better to determine the distance function, which does not depend on the direct distance between the two points (Euclidean distance), but rather we choose the distance that works on the principle that reaching a two-point does not necessarily have to be directly reached such as Canberra distance method. Ultimately, the proposed system aims to distinguish between bots and humans by employing principles of a data stream to optimize memory usage and execution speed.

### 3.5.4 Implementing the K-mean Stream Algorithm

After defining the cluster count and distance metric, we use the Stream K-Means algorithm for efficient Twitter bot detection, optimizing computational resources and memory use compared to the standard K-Means. Our system applies the Stream K-Mean clustering algorithm with a combination of functions, each with its pseudo-code, for a comprehensive overview of the process.

The first function in the Stream K-Mean clustering algorithm is called myGeneratorfun (Algorithm Implement Function "Generatorfun"). This function reads data from a CSV file in chunks of size "wsize" and yields each chunk. To achieve this, the function employs the pandas read_csv function with the chunk size parameter, which allows the CSV file to be read in chunks of a specified size. The myGeneratorfun function then converts each chunk into a numpy array and yields the array. By using the yield statement, the function returns a generator object, which can be used to iterate over the chunks of data in the CSV file. This approach is beneficial when dealing with large datasets, as it allows to process the data in smaller chunks instead of loading the entire dataset into memory.

| **Algorithm 3.5: Implement Function "Generatorfun"** | |
|---|---|
| | **Input: X - Dataset from Variance-Based Features Selection Algorithm**<br>**Output: Divide the dataset into multi-window size** |
| | Function: myGeneratorfun(wsize=2298) |
| | Input: wsize: window size (default value is 2298) |
| | Output: A generator object that yields a chunk of data |
| 1 | Initialize counter = 0 to track the number of chunks read |
| 2 | Loop through chunks of data in the Caverlee.csv file using Pandas read_csv function with chunksize=wsize |
| 3 | Convert each chunk of data to an array and store it in the chunk_array variable |
| 4 | Increment counter by 1 |
| 5 | Yield a tuple containing feature data (all columns except the last one) and target data (last column) using the chunk_array |
| 6 | If the counter is >=15, break loop |
| 7 | Return generator object |

The second function in the Stream K-Mean clustering algorithm is called find_clusters (Algorithm 3.6), and it is responsible for implementing the k-means clustering algorithm. This function takes in the data X, which represents the input data, the number of clusters no_of_clusters that formed in Algorithm 3.4, and a random seed rseed, which is an optional argument that defaults to 2 if not specified. The find_clusters function starts by randomly selecting no_of_clusters data points as the preliminary centers of a cluster. The clustering process begins with using a set of initial centers as a reference point. The next step is to update these centers iteratively by finding the data points mean within every individual cluster. In each iteration, in this algorithm, the distance between every data point and the closest cluster centroid is calculated, grounded on Euclidean, and assigned the data point to that cluster. The process keeps going until the cluster centroids are no longer moving, meaning that each data point is allocated to the closest cluster centroid and no data points switch between clusters. Once the algorithm reaches this point, it returns the final cluster centers and the corresponding labels for each data point.

**Algorithm 3.6: Implement Function "find_clusters"**

| | |
|---|---|
| | **Input: X - Implement Function "Generatorfun"** |
| | **Output: Final Centers and Centroid Labels** |
| | Function: find_clusters(X, no_of_clusters, rseed=2) |
| | Input: X: feature data |
| | no_of_clusters: Number of clusters that formed in Algorithm 3.4" Adaptive DBSCAN Algorithm " |
| | rseed: Random seed (default value is 2) |
| | Output: Tuple-containing centers and centroid labels of clusters |
| 1 | Generate initial random values for cluster centers using the numpy RandomState function |
| 2 | Loop until the centers converge: |
| 3 | To determine the smallest space between points and their respective cluster centroid, by utilizingthe pairwise_distances_argmin function. |
| 4 | To obtain the new centers, by calculating the mean of all the data points within each of the respective clusters. |
| 5 | Check for convergence by comparing the old and new centers using np.all() function |
| 6 | If the centers have converged, break the loop |
| 7 | Return the final centers and centroid labels |

The final function called "centroid_assignation" is designed to operate on a given dataset called "dset" and a set of "centroids"( **Algorithm 3.7**). The purpose of this function is to assign each data point in "dset" to its nearest centroid. To achieve this, the function calculates the error between each data point and each centroid. The function iterates through each data point in "dset" and calculates the error between that point and the centroid. It then allocates data points to the centroid with the smallest error. The calculation of the error between a data point and a centroid is performed using another function called "rsserr". Once the assignment is completed, the function returns two lists: "assigned_centroids" and "centroid_errors". The "assigned_centroids" list contains the index of the centroid to which each data point is assigned. The "centroid_errors" list contains the error between each data point and its assigned centroid.

**Algorithm 3.7: Implement Function "centroid_assignation"**

```
    Input: X - Output from Implement Function "find_clusters"
    Output: Assigns Each Observation to Nearest Centroid
    Function: centroid_assignation(dset, centroids)
    Input: dset: feature data
        centroids: Set of centroids represents the k clusters that have been identified in
       Algorithm 3.6: function "find_clusters".
    Output:    return assigned points for centroids, centroid errors
1   Get the number of centroids and observations
2   Create empty lists to store the assigned centroid index and error
3   assigned_centroids = [ ]
4   centroid_errors = []
5   Loop over all observations in the dataset:
6       Calculate the error for each centroid
7       centroid_errors_for_observation = []
8       Loop over all centroid set:
9           Getting the error between the centroid and each of the points in the dataset
10          Append the error of the dataset point in the list (centroid_error)
11          Assign the observation to the centroid with the minimum error
    Append error for all dataset points in the list (centroid_error)
7   Return the assigned centroid indices and their errors
```

Finally, the main function runs the previously mentioned functions to derive the outcome and assign a label to the dataset whether a bot or a human. Once the data have been labeled, the final step of the suggested system is initiated, which assesses the algorithm's output based on the obtained results.

### 3.6 The Evaluation Phase

It is crucial to perform a clustering evaluation to assess the accuracy and effectiveness of the produced clusters. This includes homogeneity metric, completeness metric, v-measure metric, adjusted-rand metric, adjusted mutual info metric, silhouette metric, and Fowlkes mallows metric.

### 3.7 Summary

The proposed system for detecting malicious social media bots consists of five phases, each designed to ensure efficient implementation. These phases are preprocessing, feature analysis, experiment, development, and evaluation. The detection of malicious social media bots is essential as these bots can spread harmful content, and detecting and preventing such attacks require advanced techniques such as unsupervised machine learning clustering.

Feature extraction and selection is the one crucial aspect of using machine learning methods for bot detection is clear. This is because some attributes of both human accounts and bots may appear similar, making it challenging to differentiate between the two. To overcome this issue, the system is divided into two main parts.

The first part involves preparing the essential features of Twitter accounts through three stages: data preparation, creating novel features, and identifying top-performing features while reducing computational time through PCA. The selected features are evaluated using three clustering techniques before the second part, which focuses on predicting whether a Twitter account is human or bot.

In the second part, the system utilizes integrated unsupervised algorithms, and improves three clustering algorithms  KNN, DBSCAN, and Stream K-means, to predict whether a Twitter account is human or bot. This part aims to overcome limitations that one approach alone may not address. By combining the strengths of multiple clustering algorithms, the proposed system aims to accurately detect bots using unsupervised machine learning algorithms.

In summary, the proposed system for detecting malicious social media bots is designed to ensure efficient implementation and accurate results. The system is divided into two main parts, with the first part involving preparing essential features of Twitter accounts and the second part focusing on predicting whether a Twitter account is human or bot using integrated unsupervised algorithms. The proposed system is designed to accurately detect bots using unsupervised machine learning algorithms and can be a valuable tool for preventing the spread of harmful content on social media.

# Chapter Four

## *Results and Discussion*

## 4.1 Overview

This chapter presents the results of the experiments that were conducted to evaluate the performance of the proposed system on the Caverlee dataset. The system was created to solve the problem of detecting Twitter bots, and the experiments aimed at determining its effectiveness. An overview of the Caverlee dataset and its characteristics is given first. Then, the experimental stages of the proposed system, including data preparation, model training, and performance evaluation are described. The results of the experiments are presented, and the system's performance is analyzed using various metrics, such as the Homogeneity metric, Completeness metric, V-measure metric, Adjusted Rand Index metric, Adjusted Mutual Information metric, Silhouette Coefficient metric, and FowlkesMallow metric. The implications of these results are discussed, and recommendations for future research are provided.

## 4.2 System Environment

**Hardware:** The system worked on a powerful 11th Gen Intel Core i5 processor, 16 GB RAM, and a 512 GB SSD for fast storage.

**Operating System:** Windows 11 Home 64-bit provided good performance and stability for programming needs.

**Programming Language:** Python 3.8.5 is a widely-used version of the Python programming language that provides many features and improvements over earlier versions. Version 64-bit was used to take full advantage of the system's hardware capabilities, including its RAM and processor.

**Python environment:** Spyder is an Integrated Development Environment (IDE) for Python that provides many features to help with development, such as a code editor, debugger, and data exploration tools. Spyder version 4.1.5 is relatively recent and should provide a stable and feature-rich environment for Python development. Spyder 4.1.5 and Python 3.8.5 64-bit provide a powerful and flexible platform for your Python development work, whether you're working on data analysis or machine learning.

## 4.3 Description of the Twitter Dataset

Various datasets have been utilized to assess bot detection methods. One of the primary datasets initially utilized for bot detection is Caverlee-2011 [193]. This dataset comprises Twitter data collected from December 30, 2009, to August 2, 2010. It encompasses 22,223 content polluters, including their followers' activity over time, and 2,353,473 tweets. Moreover, the dataset comprises 19,276 legitimate users, their followers' activity over time, and 3,259,693 tweets. Table 4.1 provides a summary of the Caverlee dataset for each type of Twitter account, both before and after filtering for English-language tweets. The data includes the number of user

profiles and tweets associated with each account type. The "User Profiles (Before Filtering)" column represents the total number of user profiles for each account type before applying the English language filtering. The "Tweets (Before Filtering)" column shows the total number of tweets for each account type before filtering. The "User Profiles (After Filtering)" column indicates the remaining number of user profiles after filtering for English-language tweets. Lastly, the "Tweets (After Filtering)" column displays the remaining number of tweets after the filtering process.

The decision to use this dataset for testing the proposed system is based on multiple factors. Firstly, the dataset has been widely used in various studies, and most of these studies have utilized supervised machine-learning techniques that are known for their high accuracy and quick implementation. This poses a significant challenge in evaluating the results of the proposed system, which relies on unsupervised machine learning algorithms. The second reason is that the dataset lacks specific features that can effectively differentiate legitimate accounts from malicious bot accounts. Therefore, it is crucial to test the proposed system on this dataset to evaluate its effectiveness in detecting bot accounts using unsupervised machine-learning techniques.

**Table 4.1: Caverlee Dataset Summary - Twitter Accounts Pre/Post English Tweet Filtering**

| Class | User Profiles (Before Filtering) | Tweets (Before Filtering) | User Profiles (After Filtering) | Tweets (After Filtering) |
|---|---|---|---|---|
| **Polluters** | 22,223 | 2,380,059 | 20,292 | 2,090,802 |
| **Legit Users** | 19,276 | 3,263,238 | 14,180 | 1,611,205 |

## 4.4 Structure of Data

To gain a deeper understanding of the data structure within the Caverlee dataset, the elbow method was employed as shown in Figure 4.2, which is a

commonly used technique in unsupervised learning [194]. This method utilizes the K-Means clustering algorithm to determine the optimal number of clusters for the data. By applying this technique, it was aimed to assess the impact of the selected features on the data structure.

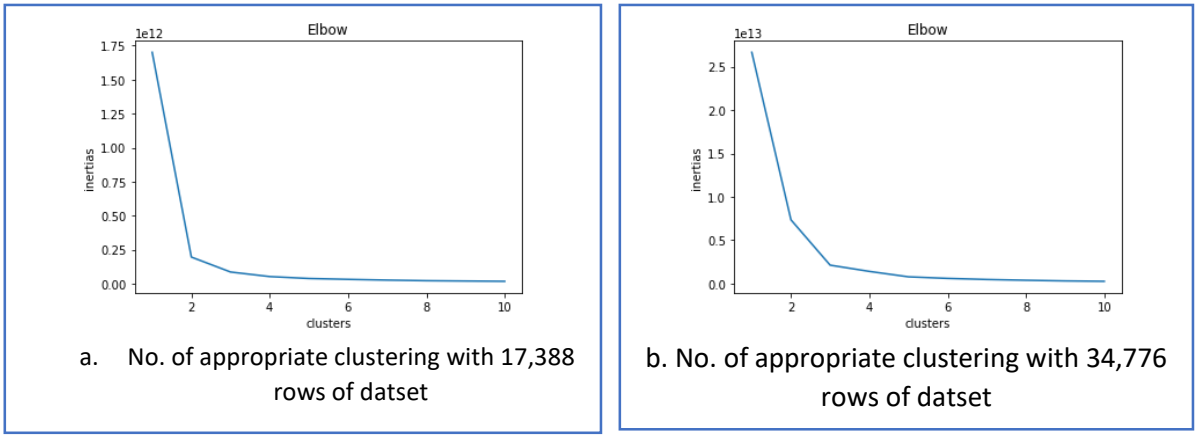
The analysis involved implementing the K-Means algorithm with different cluster numbers. Then, the performance of each clustering solution was evaluated by examining the error rate and the Silhouette Coefficient score. The error rate represents the extent to which the data points deviate from their assigned clusters, while the Silhouette Coefficient measures the degree of separation between the clusters.

After conducting the analysis, it was observed that selecting either 2 or 3 clusters resulted in the lowest error rate. Furthermore, these solutions exhibited a high Silhouette Coefficient score of 0.98, indicating well-separated clusters. This suggests that the data points within these clusters were distinct from each other, implying the presence of clear patterns or characteristics.

Based on these findings, we can infer that clustering techniques, such as K-Means, can be valuable in identifying and distinguishing bots from legitimate users. By leveraging these methods, it becomes possible to discern meaningful groupings within the data, which can aid in the detection of bots by differentiating them from genuine users.

The proposed system also analyzed the data using a dendrogram. A key element in interpreting a dendrogram is paying attention to the level at which two objects are linked. The height in the dendrogram reflects the sequence in which clusters were merged. Figure 4.3 provides a dendrogram where the heights signify the distances between clusters. The dendrogram demonstrates that using origin

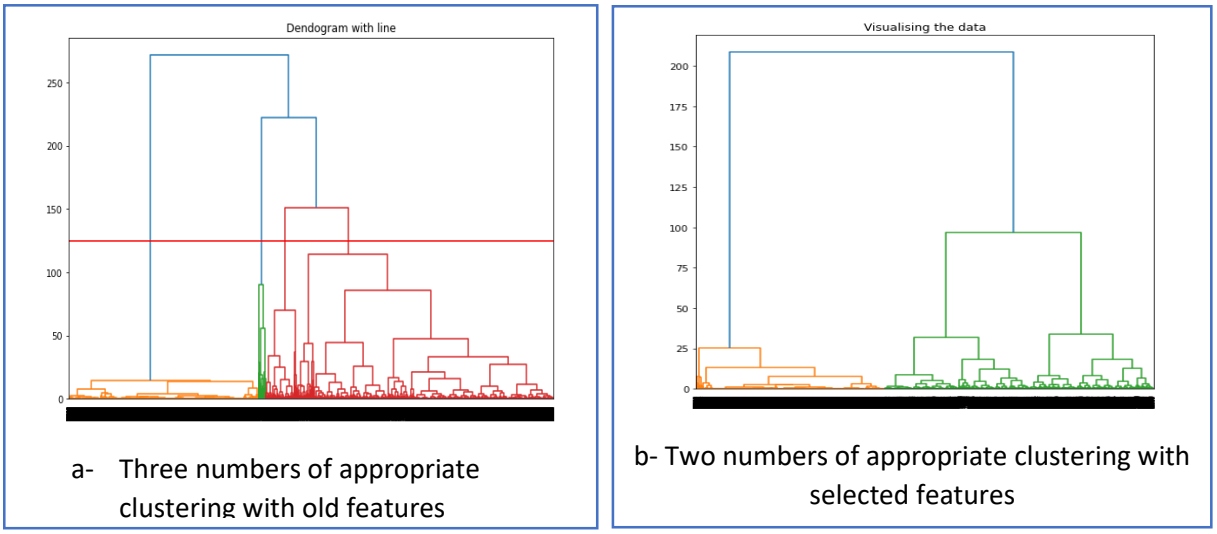
features (Figure 4.3.a.) results in a different number of clusters compared to using proposed features (Figure 4.3.b.).




a. No. of appropriate clustering with 17,388 rows of datset

b. No. of appropriate clustering with 34,776 rows of datset

**Fig.4.1: Elbow Method For Optimal Cluster Numbers Selection**




a- Three numbers of appropriate clustering with old features

b- Two numbers of appropriate clustering with selected features

**Fig.4.2: Hierarchical Dendrogram for Clusters and Subclusters**

To verify our feature selection and enhance the comprehensiveness of our data analysis, we employed three additional techniques: Distortion Score, Calinski Harabasz, and Silhouette Score. We created three plots (Figures 4.4, 4.5, and 4.6) that demonstrate the correlation between the obtained scores and the number of clusters generated by two types of clustering algorithms. Part (a)

of the plots depicts the application of these methods on agglomerative clustering, while part (b) shows their application on k-means clustering. The curve on the plots resembles an elbow, with the point of inflection indicating the optimal number of clusters for the given dataset. This information provides valuable insight into the nature of the data and supports the effectiveness of our selected features for the proposed system.

Figure 4.4 demonstrates that the scoring parameter metric is set to distortion, which computes the sum of squared distances from each point to its assigned center. However, two other metrics can also be used with the calinski_harabasz as in Figure 4.5 and the KElbowVisualizer – a silhouette as shown in Figure 4.6. The silhouette score calculates the mean Silhouette Coefficient of all samples, while the calinski_harabasz score computes the ratio of dispersion between and within clusters. The amount of time to train the clustering model per K is represented as a dashed orange line, whereas the other line (blue line) represents the error value for points.

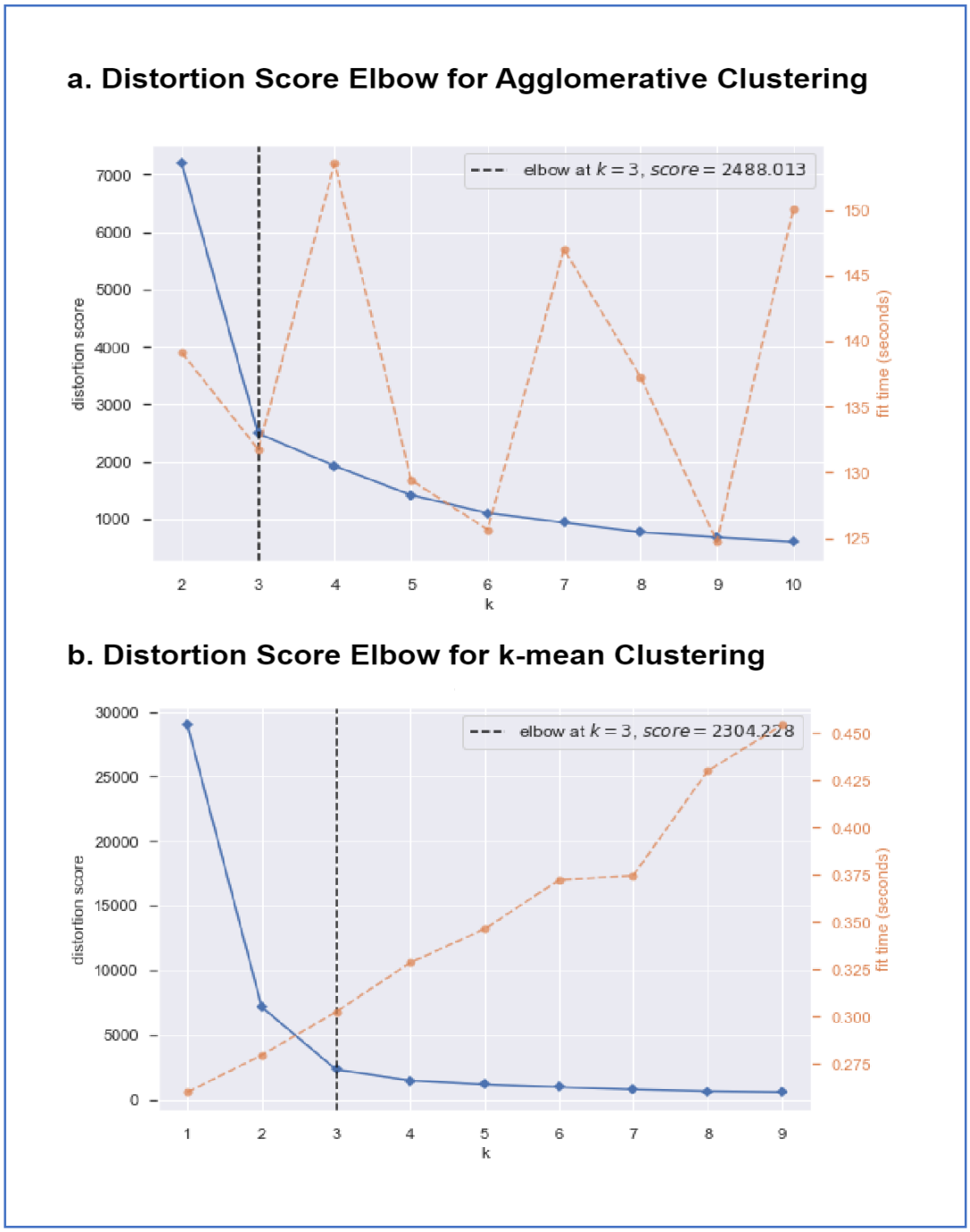



**Figure 4.3: Distortion Score Elbow Plot**

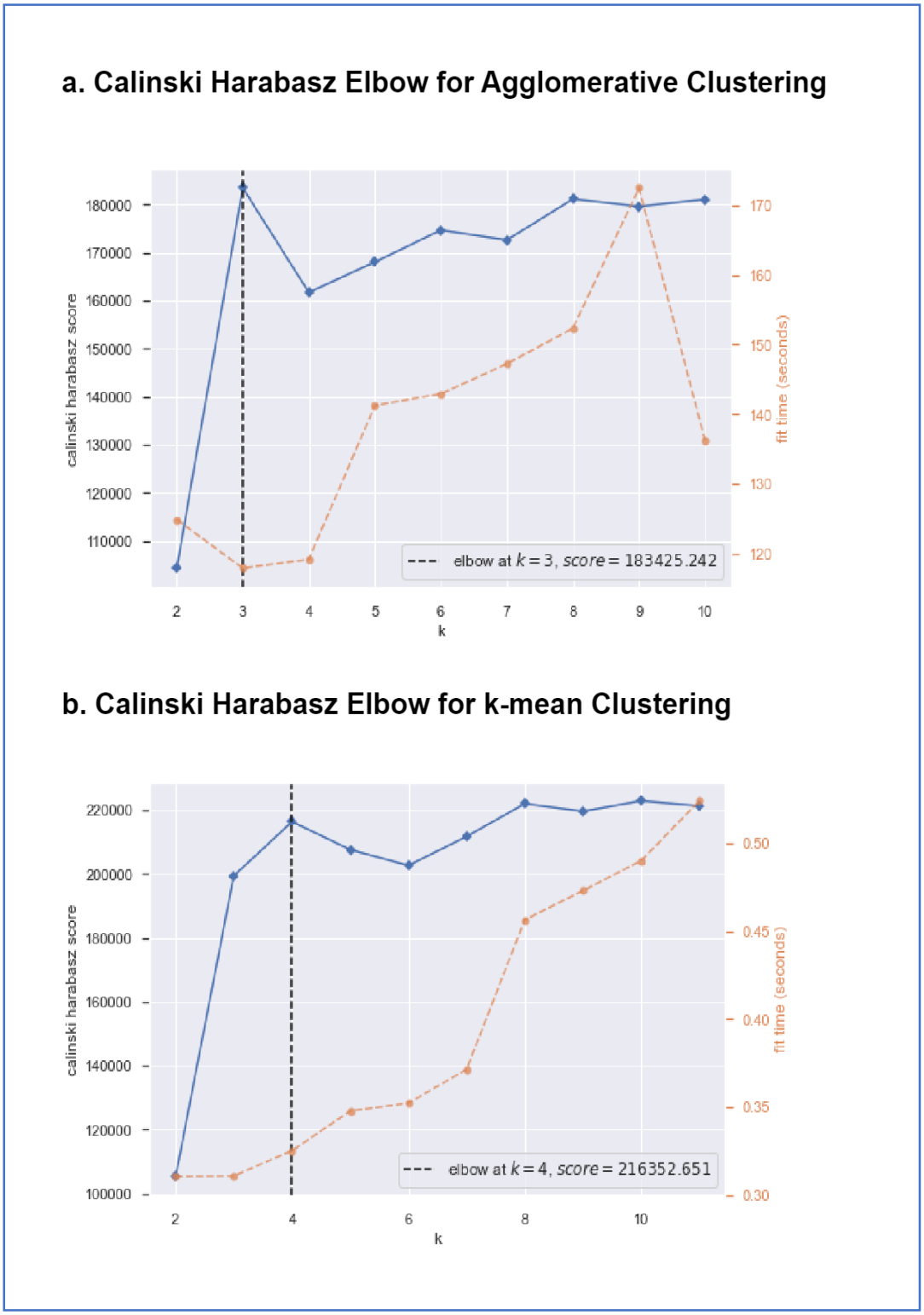
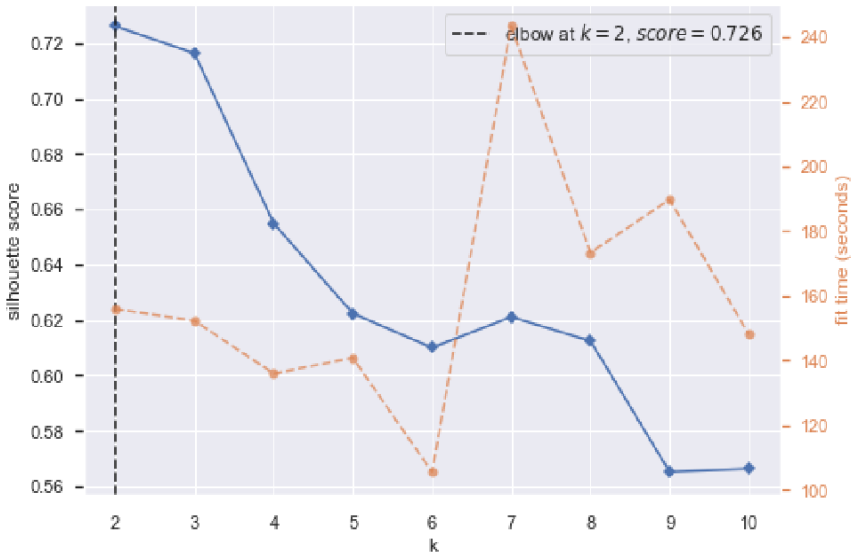



**Figure 4.4: Calinski Harabasz Score Elbow Plot**

a. Silhouette Score Elbow for Agglomerative Clustering




b. Silhouette Score Elbow for k-mean Clustering

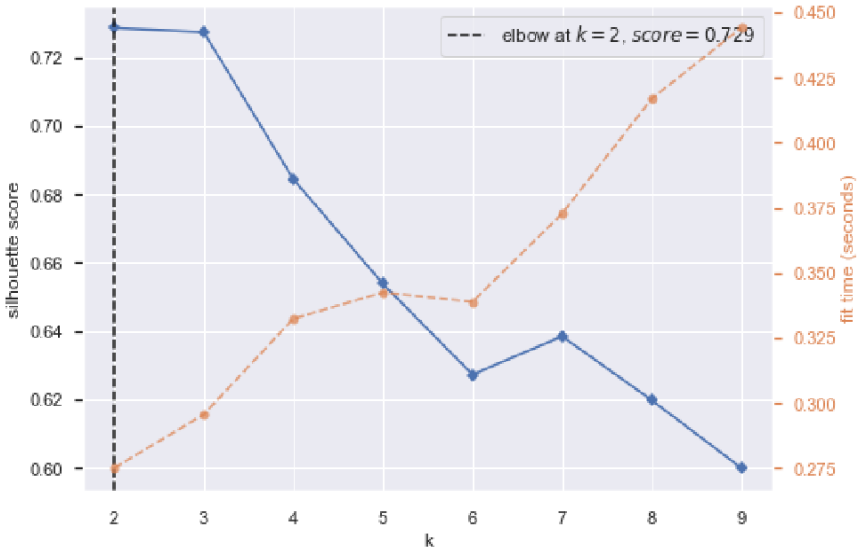



**Figure 4.5: Silhouette Score Elbow Plot**

## 4.5 Features Ranking

One advantage of using the Correlation Attribute Eval (CAE) technique for feature selection is that it produces a ranking of the features based on their correlation with the target variable [195]. This can be useful in identifying the most important features of a given problem and prioritizing them for further analysis. By using the ranking provided by CAE, it is possible to quickly narrow down the set of relevant features and improve the efficiency of the feature selection process. The ranking is based on the correlation between each attribute and the output variable, with features having the highest correlation ranked at the top. Table 4.2 presents the results of the CAE method, which ranks the attributes based on their correlation with the target variable. It serves as a valuable reference for assessing the relative importance and correlation of each attribute based on the CA method. This information allows us to gain insights into the relationship between these attributes and the target variable in the analysis. In the "Index" column, the numerical index represents the position of each attribute in the ranking list. The "Ranked attributes" column displays the correlation values assigned to each attribute by the CAE method. Higher values indicate a stronger positive correlation with the target variable, while lower values indicate a stronger negative correlation. The "Attribute name" column provides the names or descriptions of the attributes being evaluated. Each attribute corresponds to a specific characteristic or measurement related to the data under analysis.

Attributes ranked higher in the list, such as "FollowingToAgeRate" at index 11 and "max-min" at index 18, demonstrate a stronger positive correlation with the target variable. On the other hand, attributes ranked lower in the list, including "NumberOfTweets" at index 10 and "ProFollowRatio" at index 9, exhibit a stronger negative correlation.

When utilizing a positive cutoff for determining relevant attributes, removing the attributes that exhibit a negative correlation can be considered. By doing so, the focus was on the attributes that have a stronger positive correlation with the target variable, which are likely to provide more meaningful insights and contribute to the analysis effectively.

**Table 4.2: Ranking of attributes concerning the Correlation Attribute Eval (CA) method**

| Index | Ranked attributes | Attributes name |
|---|---|---|
| **11** | 0.39513 | FollowingToAgeRate |
| **18** | 0.36396 | max-min |
| **17** | 0.35699 | CV_Following |
| **5** | 0.20985 | NumerOfFollowings |
| **1** | 0.16617 | LngthOfScreenName |
| **8** | 0.13545 | FollowingtoFollowerRatio |
| **2** | 0.12405 | LengthOfDescriptionInUserProle |
| **12** | 0.07547 | FollowerToAgeRate |
| **6** | 0.03236 | NumberOfFollowers |
| **14** | 0.00335 | AvarageDiffrenceTweetRatio |
| **20** | 0.00323 | bfr-afr |
| **7** | -0.01787 | FollowertoFollowingRatio |
| **19** | -0.05244 | CV_Months |
| **15** | -0.09551 | MaxOfDays |
| **13** | -0.1005 | No_oftweets/accountage(month) |
| **16** | -0.10284 | MaxOfMonth |
| **10** | -0.1235 | NumberOfTweets |
| **3** | -0.18584 | age_month |
| **4** | -0.1913 | age_days |
| **9** | -0.58003 | ProFollowRatio |

## 4.6 Scrutinizing Quality of the Selected Features

The investigation discussed focuses on the evaluation of the effectiveness of cluster algorithms in detecting bots with the help of proposed features. The proposed system has implemented three types of cluster algorithms: partition, hierarchy, and density. These algorithms are represented by four different techniques, namely k-mean, k-medoid, agglomerative, and DBSCAN. The system used the original features initially, and then the proposed features to evaluate the model using the four

techniques mentioned above. The results obtained from each algorithm are then compared to determine which one obtained the highest accuracy. The outcomes of this method are presented in Figure 4.7 which shows the final results of four cluster algorithms. Part "a" displays the outcomes obtained using the original features, while part "b" shows the results obtained using the proposed features. The results indicate that the proposed feature selection improved the performance of the cluster algorithms compared to the original features. The use of the proposed features led to better cluster separation and a higher level of clustering accuracy.

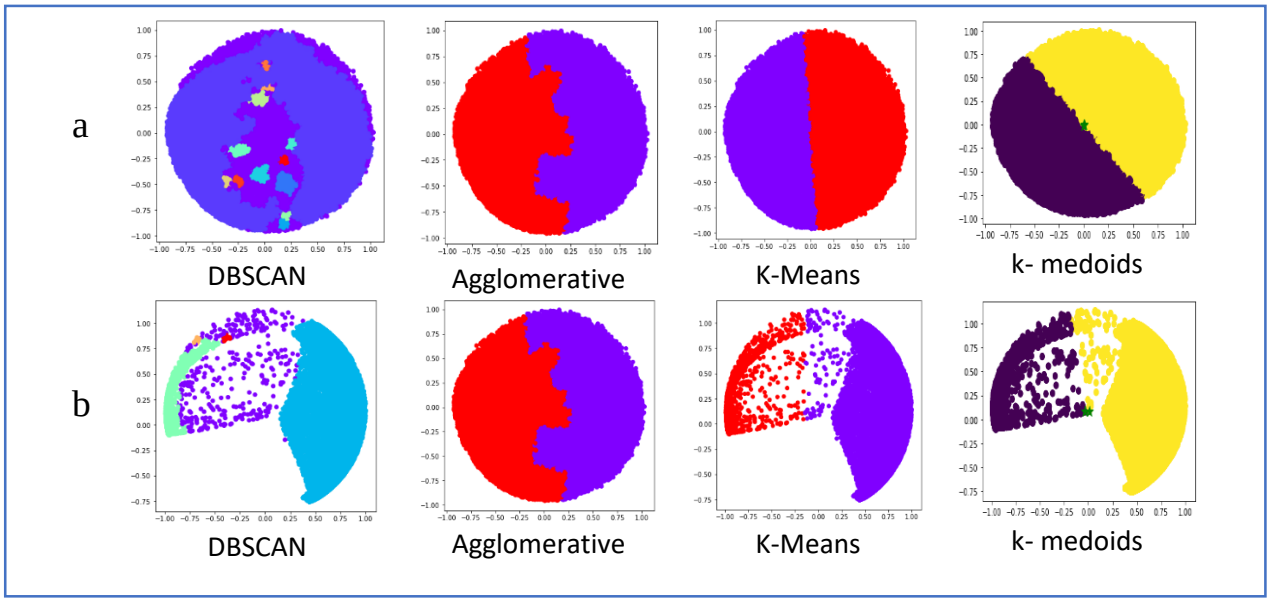



**Figure 4.6: Comparison of clustering results using original and proposed features**

The results presented in Table 4.3 demonstrate the efficacy of the proposed features in enhancing the accuracy of the four cluster algorithms employed in this dissertation. The main result indicates that the proposed features have surpassed the original features in terms of accuracy, with a minimum improvement of 0.306 and a maximum improvement of 0.471 across the algorithms. The improvement range was calculated by deducting the original accuracy score from the accuracy score achieved through the proposed features for each algorithm. For instance, the

Agglomerative algorithm achieved an accuracy improvement of 0.471, as calculated by subtracting 0.525 from 0.996. Similarly, the k-medoids algorithm achieved an improvement of 0.306, DBSCAN an improvement of 0.378, and K-Means an improvement of 0.469. Therefore, the range of improvement obtained by utilizing the proposed features was between 0.306 and 0.471.

**Table 4.3: Fowlkes-Mallows Scores for Selected Cluster Algorithms Accuracy**

| clusteringAlg. | Alg. Parameters | Fowlkes mallows scoreWith Origin Features | Fowlkes mallows scoreWith Proposed Features | Range Of Improvement |
|---|---|---|---|---|
| Agglomerative | n_clusters = 2 , affinity='euclidean', linkage='ward' | 0.525 | 0.996 | 89.71% |
| k- medoids | starting_medoids=randome,k=2 | 0.591 | 0.897 | 51.78% |
| DBSCAN | eps = 0.0395, min_samples = 50 | 0.613 | 0.991 | 61.67% |
| K-Means | n_clusters=2, init='k-means++', random_state=42 | 0.525 | 0.994 | 89.33% |

In summary, our findings show that the proposed features outperform the original ones in bot detection. These results can benefit the enhancement of bot-detection techniques and the development of more efficient bot-detecting models. The equation to calculate the percentage increase from 0.306 to 0.471 is:

**[(New Value - Old Value) / Old Value] x 100%** **(4.1)**

Substituting the values, we get: [(89.71 - 51.78) / 51.78] x 100% = 73.27 Therefore, the percentage increase from 51.78 to 89.71 is approximately 73.27.

## 4.7 Comparing the proposed approach with the supervised techniques

The presented findings in Table 4.4 demonstrate a significant improvement in the accuracy of four cluster algorithms for identifying bots, achieved through the

proposed methodology. The inclusion of the proposed features in the original training data consistently enhances the performance of cluster algorithms. Additionally, the proposed approach showed almost a 10% improvement in unsupervised learning algorithms compared to supervised techniques utilized in the previous study. Despite supervised techniques' higher accuracy and faster processing speed due to the availability of the label, the proposed approach showed comparable accuracy improvement in unsupervised learning as shown in Table 4.4.

The results presented in the dissertation demonstrate the valuable information contained in the generated features, which greatly aided the unsupervised machine learning algorithms in effectively distinguishing between bot accounts and human accounts. It is worth noting that these algorithms achieved this without the reliance on pre-categorized data, as is typically required by supervised machine learning algorithms. Furthermore, the four clustering algorithms not only exhibited the ability to differentiate between accounts, but they also surpassed the accuracy of the supervised learning algorithms. This outcome highlights the strength of the clustering approach in this context.

One of the key challenges faced when using supervised machine learning for social bot detection is the limited generalizability of models trained on specific datasets. This limitation becomes particularly pronounced when encountering new data or updated versions of bots. In such cases, the characteristics of the new versions may differ from those the model was originally trained on, posing significant challenges.

Additionally, determining appropriate sizes for the training and testing sets presents another challenge in machine learning. If the training set is too small, the algorithm may not have sufficient data to effectively learn. Conversely, an insufficiently sized validation set can result in substantial variance in accuracy,

precision, recall, and F1 score. Furthermore, ensuring that the test data set follows the same probability distribution as the training data set is a crucial consideration. Overcoming these challenges is vital to mitigate bias issues that arise, particularly when novel bot versions are introduced.

The positive outcome of the aforementioned comparison is an indication of the success of the first phase of our proposed system, which involves generating new features. These features have played a crucial role in enabling the system to accurately distinguish between accounts without the need for pre-categorized data.

However, it is important to acknowledge that clustering algorithms are not without drawbacks. One such drawback is their dependence on input parameters, which directly impact the accuracy of their results. As previously mentioned, the determination of these parameters typically involves a trial-and-error approach, necessitating time and resources for implementation. Consequently, the second part of the proposed system is designed to address this issue, aiming to achieve a fully automatic bot detection process. Further details regarding this aspect are provided.

**Table 4.4: Results of a Previous Supervised Study on the Same Dataset**

| Ref | Technique | Type | Accuracy |
|---|---|---|---|
| [5] | RandomForest algorithm | supervised | 95 |
| [193] | Random Forest Accuracy | supervised | 98.42 |
| [196] | deep-learning model | supervised | 95 |
| [197] | Random Forest | supervised | 57.139 |
| | **The proposed method** | **unsupervised** | **99.86** |

## 3.8 Feature Correlation Analysis

Feature Correlation Analysis is a technique in data science that helps to discover the relationships between different variables in a dataset. It is an essential method used to gain valuable insights and make informed decisions. Analyzing

correlations between features helps researchers understand how variables are related and how they may impact the outcome of an analysis or model. Here, the findings of the correlation analysis for the selected approach features are explored using different types of correlation analysis techniques.

1) Pairs plot (scatterplot matrix): A pairplot is a useful visualization tool for exploring relationships between multiple pairs of variables. It allows for identifying potential strong relationships between two variables by visualizing their distributions and relationship. Figure 4.8 illustrates the relationships between the five highest-ranking features using a pairplot. To make the pairplot even more valuable, Figure 4.9 colors the figures based on a categorical variable such as a bot (1) or human (0). This allows us to explore the relationships between pairs of variables while also highlighting differences between categories.

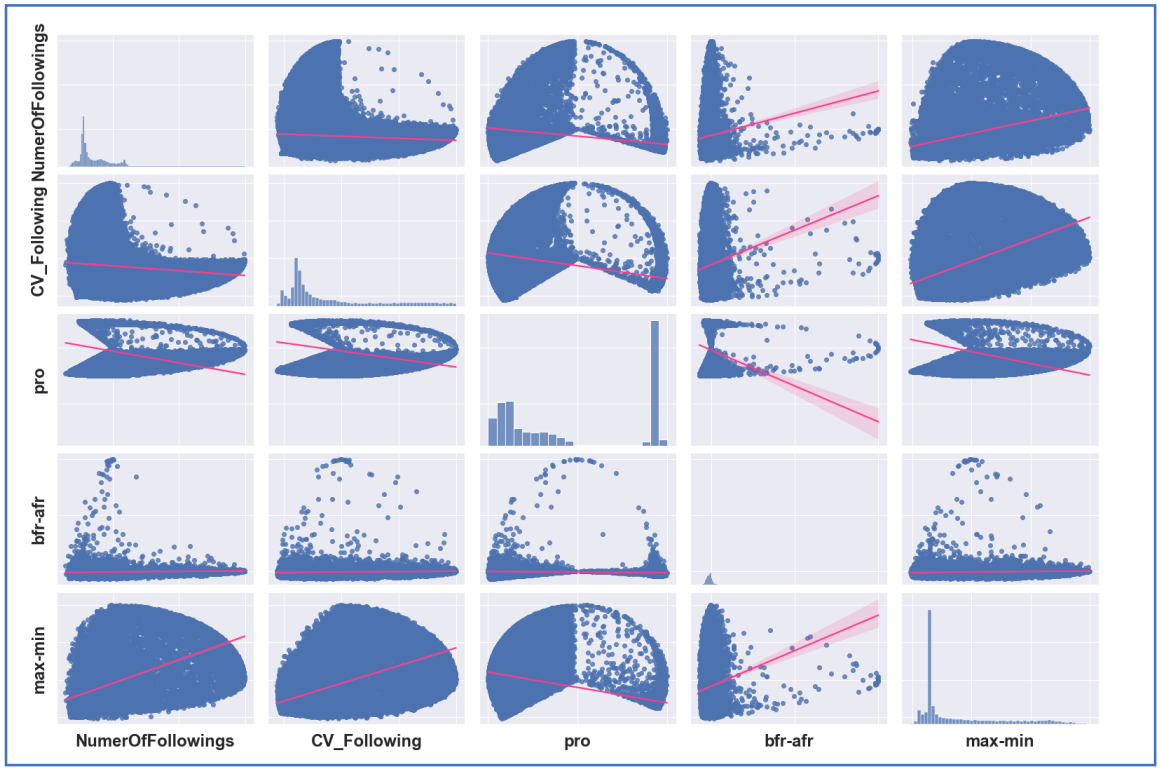



**Figure 4.7: Correlation between selected features**

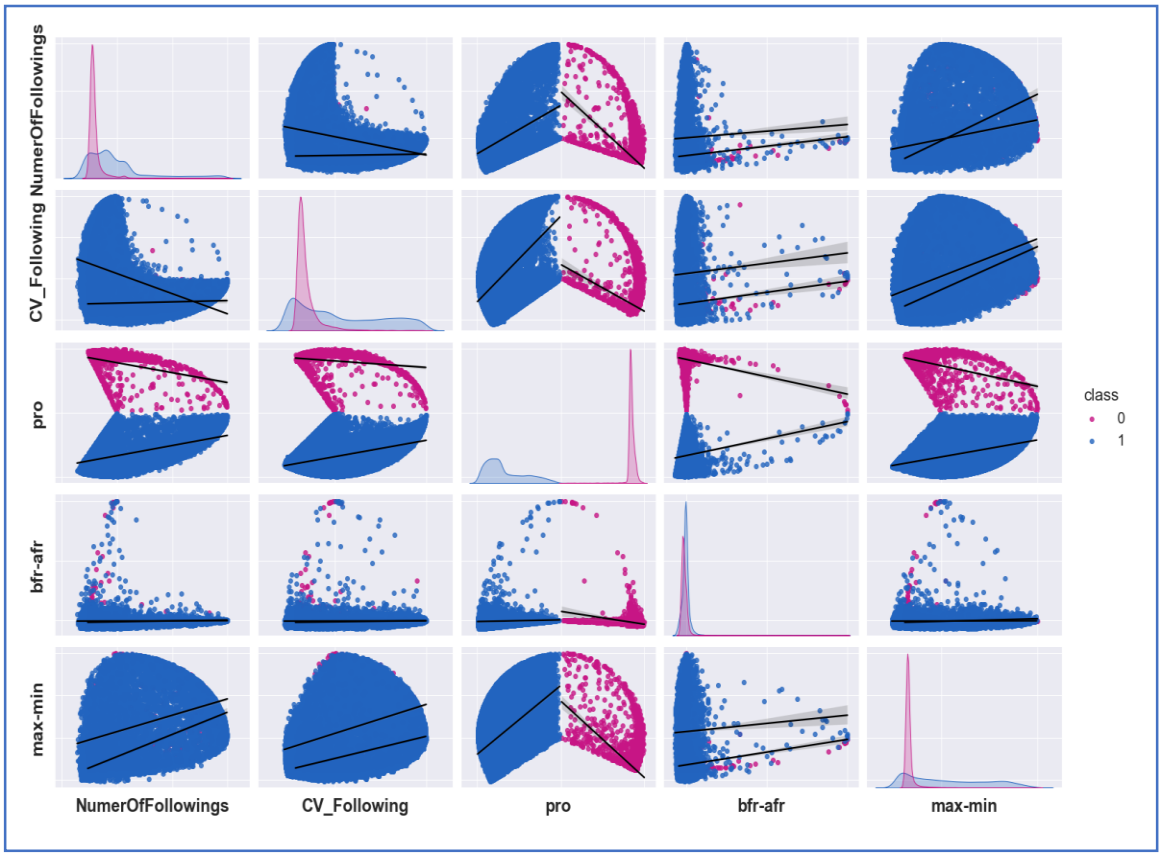



**Figure 4.8: Pairplot with a color-coded categorical variable for bots and humans**

In the process of conducting a thorough analysis of Figures 4.8 and 4.9 to enhance our understanding of the relationship between the variables, the inclusion of a pairplot can be proven to be immensely advantageous. When scrutinizing these figures, it is crucial to be attentive to the following patterns:

i) Positive correlation: When two variables have a positive relationship, you can observe a diagonal line running from the lowermost left angle to the scatterplot's upper right angle. This shows that when one variable increases, the other variable tends to increase as well. The stronger the positive relationship, the tighter the cluster of points around the diagonal line. For example, Figures 4.8 and 4.9 reveal a robust positive association between NumberOfFollowings with max-min and CV_Following with the max-min.

ii) Negative correlation: When two variables have a negative relationship, you can observe a diagonal line running from the upper left angle to the scatterplot's lowermost left angle. This shows that when one variable increases, the other variable tends to decrease. The stronger the negative relationship, the tighter the cluster of points around the diagonal line. For example, Figures 4.8 and 4.9 reveal a negative correlation between NumberOfFollowings with pro and CV_Following with pro.

iii) Non-linear relationship: If there is a non-linear relationship between two variables, the scatter plot will show a curved or U-shaped pattern, rather than a straight line.

iv) No relationship: If there is no relationship between two variables, the scatter plot will show a random scattering of points with no discernible pattern.

2) A correlation heatmap is a graphical representation of a correlation matrix illustrated in Figure 4.10, which shows the relationship between variables in a dataset. It is a valuable instrument for representing the intensity and orientation of the connection between two or more variables.

A correlation matrix displays the correlation coefficients, which are values between -1 and 1 showing the magnitude and orientation of the association between two variables. Positive coefficients indicate a positive relationship, while negative coefficients indicate a negative relationship. If the correlation coefficient is 0, it indicates that there is no correlation between the two variables. This means that there is no statistical relationship or association between them. Heatmap is created by plotting the matrix using a color scale to represent the strength of the correlation between each pair of variables.


**Figure 4.9: Heatmap Showing Correlation Between Selected Features**

## 4.9 Implementation and Experimental Results of the Modified Algorithms

This section presents the results of implementing the three modified clustering algorithms. The aim is to fully automate the process of distinguishing Twitter accounts between bots and human accounts. The modifications made to these algorithms address prominent problems in clustering and have resulted in significantly improved accuracy and faster implementation. The contributions of these modifications are discussed in detail, highlighting their significance in overcoming the challenges of clustering Twitter accounts and enhancing the performance of the proposed system.

### 3.9.1 The Adaptive KNN algorithm

By using both dimensionality reduction and a K-D Tree, the time complexity of KNN can be reduced to O (DNlog (N)), making it more practical for large datasets and high-dimensional data. The proposed system utilized the NearestNeighbors unsupervised algorithm to compute the mean distance between each point and its

n_neighbors, to obtain the optimal epsilon parameter for the DBSCAN algorithm. One parameter that needed to be specified was n_neighbors(Number Of Neighbors(K)), which was determined by applying Equation 3.18, and the resulting values were empirically validated as shown in Table 4.5 with multiple values for the number of dataset categories. The proposed system dataset consisted of three categories. The optimal value for epsilon is determined by identifying the "crook of the knee" or the point of maximum curvature, which represents the point where the additional cost of increasing epsilon no longer results in significant benefits.

**Table 4.5: Identification of "Crook of the Elbow" for Optimal Epsilon in Improved Neighbors Algorithm with Equation (3.18)**

| **Dataset Length = 34476** | | |
|---|---|---|
| **Numbers of Categories** | Number Of Neighbors(K) $\frac{\left(\sqrt[2]{\text{DatasetLengh}}\right)}{\text{Categories Type}}$ | The knee Curvature |
| **1** | 185 | 0.07 |
| **2** | 92 | 0.05 |
| **3** | 61 | 0.04 |
| **4** | 46 | 0.03-0.04 |
| **5** | 37 | 0.03 |
| **6** | 30 | 0.027 |
| **7** | 26 | 0.026 |
| **8** | 23 | 0.024 |
| **9** | 20 | 0.02 |

Based on Table 4.5, the optimal value for epsilon is approximately 0.04, with 61 neighbors, calculated using the equation and divided by the number of categories (3) in the proposed system dataset. In section 3.9.2, all the values in the "knee Curvature" column of Table 4.5 were tested with the DBSCAN algorithm to confirm that the value of 0.04 is indeed the optimal value for the epsilon parameter.

Figure 4.11 displays knee curves obtained by varying the number of neighbors in the k-nearest neighbor algorithm. The range of neighbors considered spans from 20 to 185. The knee curves indicate the change in direction of the curve, and the corresponding knee curvature values range from 0.02 to 0.07, representing the slope at the bending point. Specific knee bending points and their corresponding knee curvatures for each number of neighbors are provided in Table 4.5. These knee-bending points mark the positions on the curve where the transition occurs, indicating a change from a low to a high slope [198], [199]. The knee curvature at these points represents the optimal value of epsilon for the DBSCAN algorithm, which can be used to identify clusters in data.

By analyzing the knee bending points and their knee curvatures, the optimal epsilon parameter for the DBSCAN algorithm can be determined based on knee curvature. This parameter selection is crucial for accurate clustering. It is worth noting that DBSCAN is a density-based clustering algorithm that relies on knee curvature to find an appropriate epsilon value.

In the next section, the optimal knee curvature value is demonstrated which is 0.04 and the optimal number of neighbors is 61 when it is applied to the modified DBSCAN algorithm. These results can guide the selection of appropriate parameters for the DBSCAN algorithm, leading to improved clustering performance on specific datasets.

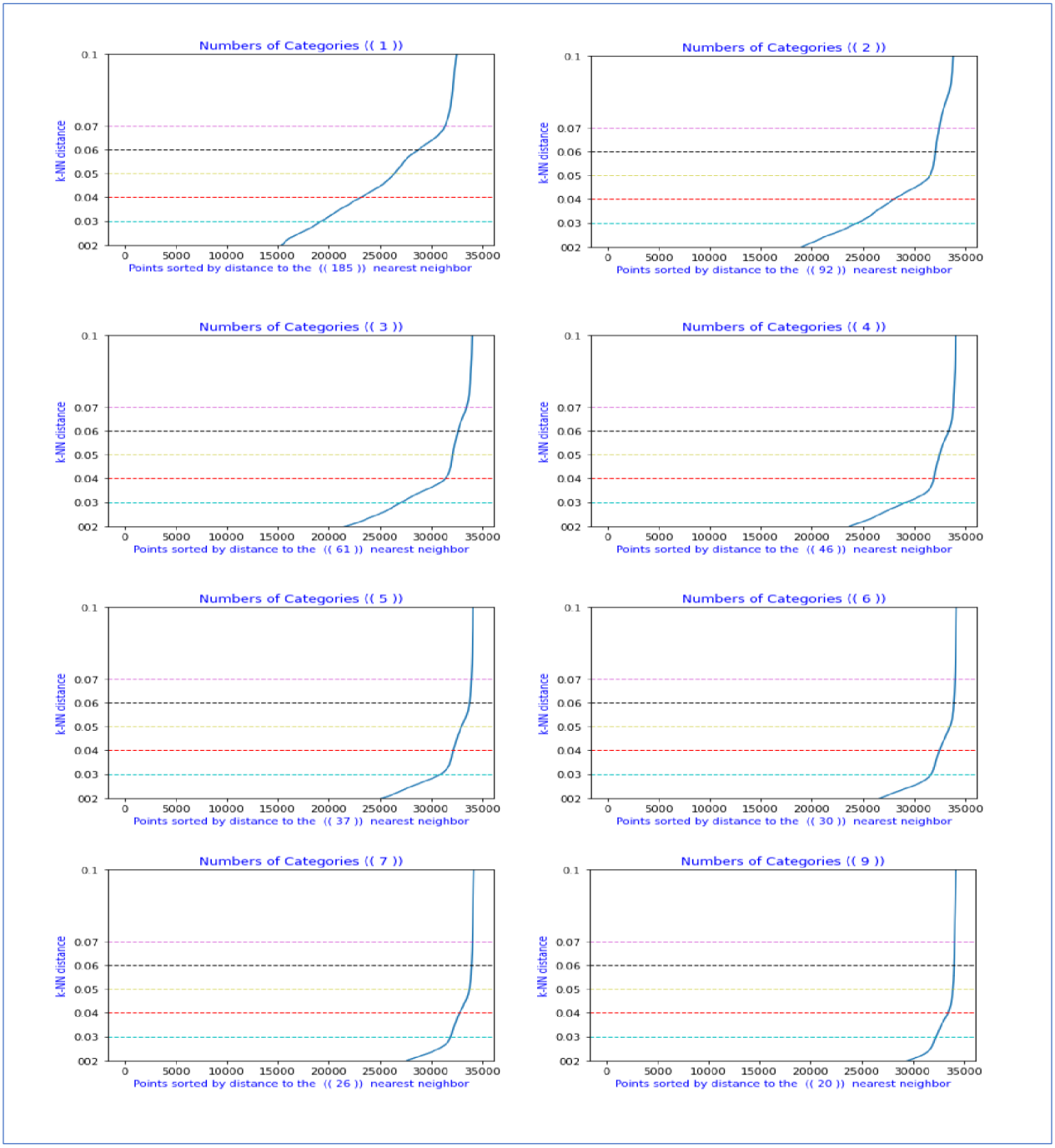



**Figure 4.10: Finding Optimal Epsilon for Improved Clustering**

## 4.9.2 The Adaptive DBSCAN Algorithm

DBSCAN can be adapted to solve problems in the k-means algorithm, such as automatic cluster determination and better distance equation selection for centroid

determination. DBSCAN requires two significant parameters to be specified, namely, Epsilon (ε) and the minimum number of points (MinPts) necessary to form a cluster. While the calculation of Epsilon was explained in the preceding section, the estimation of the minimum number of points required to form a cluster is determined using Equation 3.19.

Figure 4.12 displays both the actual labels of the dataset and the labels predicted by the DBSCAN algorithm. However, it is essential to establish a correspondence between these labels before interpreting the results. Data matching between the original labels and the predicted labels from the DBSCAN algorithm can be performed to establish a correspondence between them. This matching process involves comparing the labels assigned by the algorithm to the actual labels of the dataset. By performing this data-matching process, the accuracy and effectiveness of the DBSCAN algorithm in assigning labels to the data points can be evaluated. It allows for assessing the agreement or discrepancy between the original labels and the predicted labels, providing insights into the clustering results and the algorithm's performance.

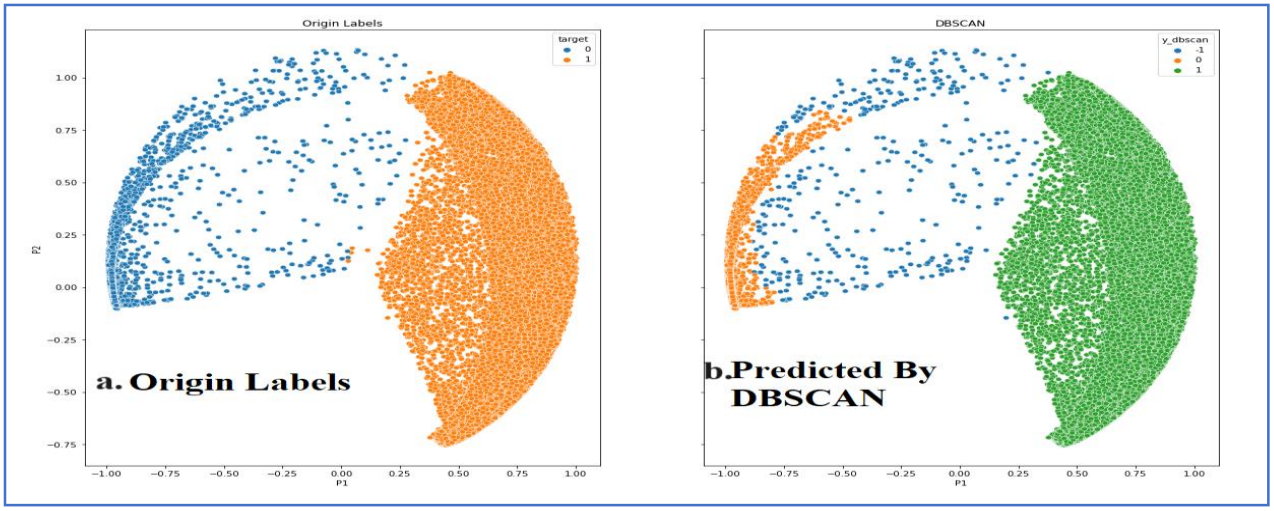



**Figure 4.11: Actual and Predicted Labels by DBSCAN Algorithm**

The results obtained from the DBSCAN algorithm are significant, as it effectively separated the data points into two distinct clusters using the proposed system features. This is demonstrated in Figure 4.13, where Cluster 0 contained 13,819 points, while Cluster 1 contained 20,286 points. Additionally, the algorithm identified 371 points as outliers, as illustrated in Figure 4.13, Cluster -1. These findings suggest that the algorithm parameters proposed by the system were effective in automatically operating the algorithm without requiring manual intervention or trial and error in parameter selection.

The results obtained from the implementation indicate that the DBSCAN algorithm is a highly efficient tool for conducting cluster analysis on extensive datasets. It effectively grouped a considerable number of data points, thereby demonstrating its capability and reliability. To evaluate its performance, the algorithm was assessed using seven similarity metrics (Homogeneity metric, Completeness metric, V-measure metric, Adjusted Rand Index metric, Adjusted Mutual Information metric, Silhouette Coefficient metric, and Fowlkes-Mallows score), as presented in Table 4.6. Impressively, the algorithm achieved high scores in all metrics: 0.998 for the Homogeneity metric, 0.930 for the Completeness metric, 0.963 for the V-measure metric, 0.983 for the Adjusted Rand Index metric, 0.963 for the Adjusted Mutual Information metric, 0.782 for Silhouette Coefficient metric, and 0.991 for Fowlkes-Mallows score. Moreover, the high scores across all metrics indicate the algorithm's accuracy and reliability in separating data, showcasing its competence in clustering tasks. Additionally, comparing the accuracy of the algorithm to that of supervised learning algorithms reveals that DBSCAN is a competitive alternative, despite its unsupervised nature.

Finally, the DBSCAN algorithm estimated the number of clusters as two, which can be used as a parameter in the Stream K-means algorithm, representing the

number of clusters. This estimate is essential in optimizing the K-means algorithm's performance in clustering the data points.

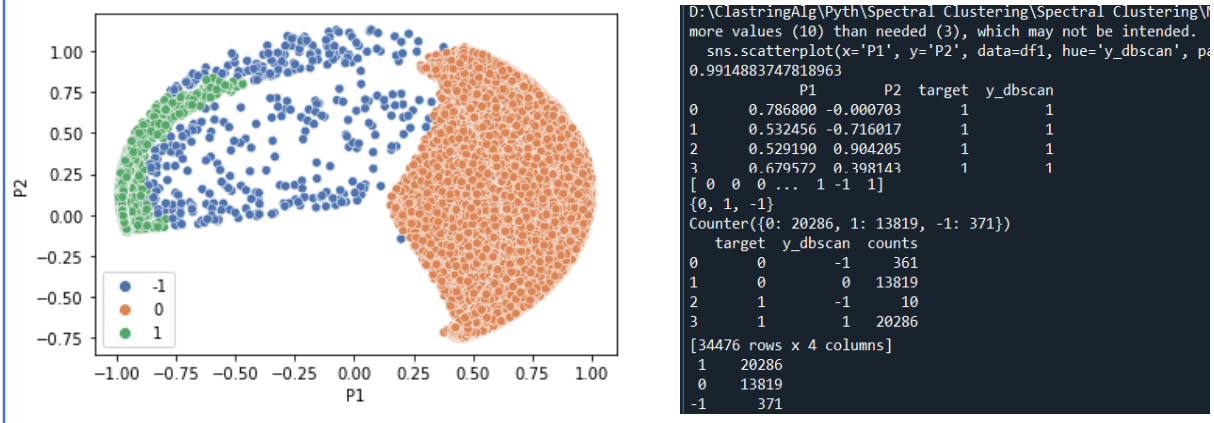



**Figure 4.12: Matching Process between Actual and Predicted Labels by DBSCAN Algorithm**

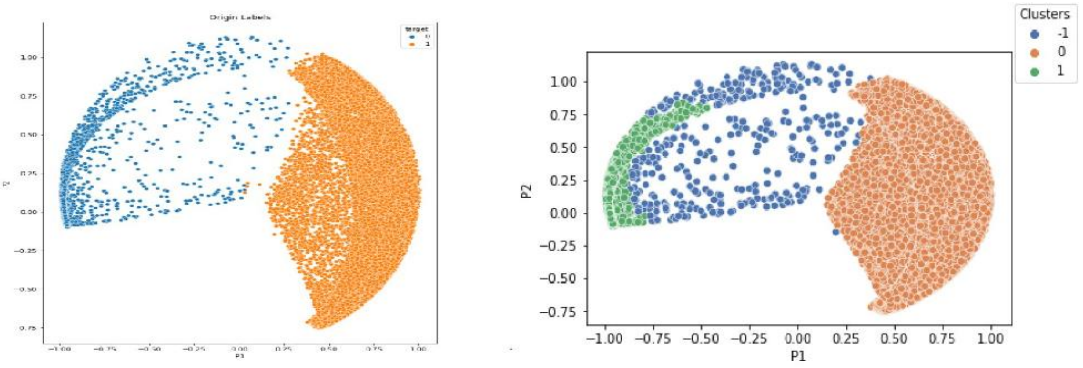
**Table 4.6: The output of the Adaptive DBSCAN Algorithm Implementation**

| DBSCAN Algorithm Output | Value |
|---|---|
| **Estimated cluster numbers** | 2 |
| **Points numbers estimated in Cluster 0** | 13819 |
| **Number of points estimated in Cluster 1** | 20286 |
| **Estimated number of noise points** | 371 |
| **Homogeneity** | 0.998 |
| **Completeness** | 0.930 |
| **V-measure** | 0.963 |
| **Adjusted Rand Index** | 0.983 |
| **Adjusted Mutual Information** | 0.963 |
| **Silhouette Coefficient** | 0.782 |
| **Fowlkes mallows score** | 0.991 |

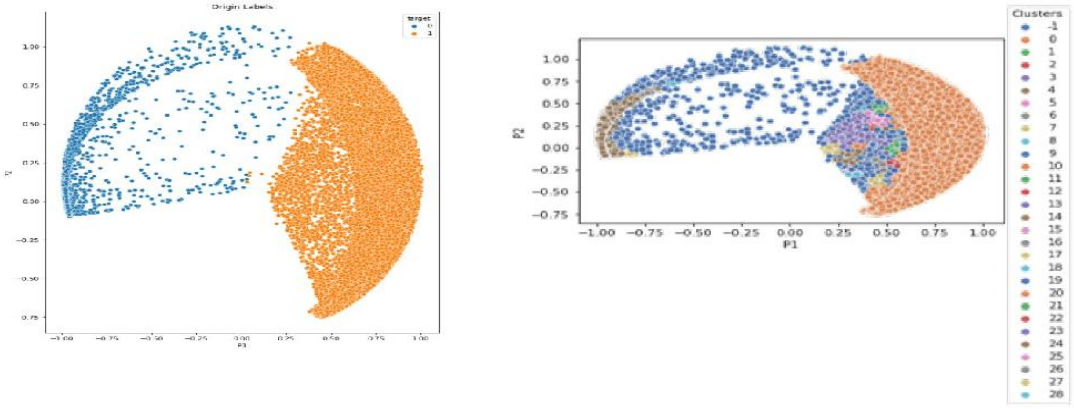
The results presented in Figure 4.14 and Table 4.7 demonstrate the impact of different values of epsilon on the performance of the DBSCAN algorithm. It is observed that increasing the value of epsilon reduces the number of clusters and noise points detected while increasing the Silhouette Coefficient and Fowlkes Mallow's score. However, the completeness score decreases as the value of epsilon increases. Based on the results, the best performance of the algorithm was achieved at an epsilon value of 0.04, which supports the proposed hypothesis when used in the proposed equation 3.18 can be used to determine the optimal value of epsilon for the given dataset.

It is also essential to note that the proposed Equation 3.18 can help determine the optimal value of epsilon for a particular dataset. This approach can save a considerable amount of time and effort that would otherwise have been required to experiment with various epsilon values manually. This method can also help eliminate any bias in the selection of epsilon values, which may arise due to the researcher's subjective opinion or prior assumptions.

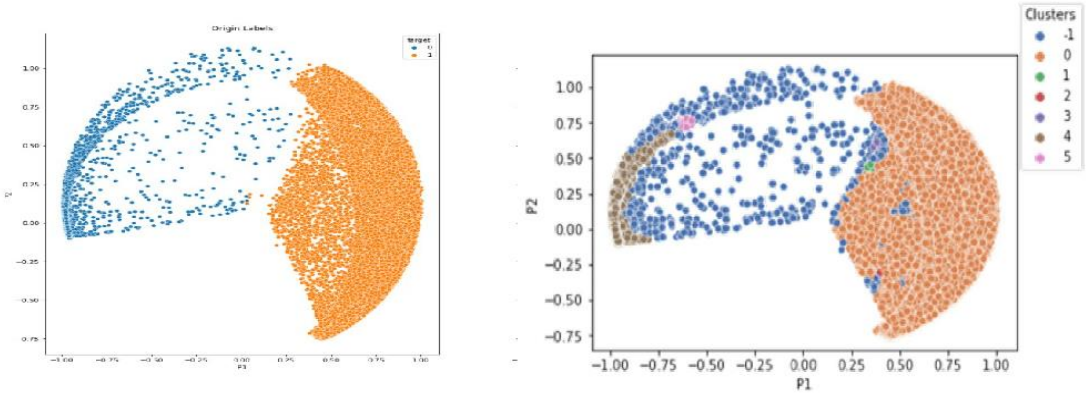
In conclusion, the proposed system findings reinforce the importance of carefully selecting the epsilon value in the DBSCAN algorithm to achieve accurate and meaningful clustering results. The proposed Equation 3.18 provides a systematic and objective approach to determining the optimal epsilon value, which can save time and improve the quality of clustering results.




a. Investigating the Performance of DBSCAN Algorithm Using 0.4 Curve Values




b. Investigating the Performance of DBSCAN Algorithm Using 0.28 Curve Values




c. Investigating the Performance of DBSCAN Algorithm Using 0.03 Curve Values

**Figure 4.13: Performance of DBSCAN Algorithm with Varying Epsilon Values (0.04, 0.28, and 0.03).**

**Table 4.7: Impact of Epsilon Values on DBSCAN Algorithm Performance**

| Epsilon Value | Estimated cluster numbers | Noise points estimated number | Silhouette Coefficient | Fowlkes mallows score | Completeness |
|---|---|---|---|---|---|
| **0.028** | 29 | 903 | 0.640 | 0.689 | 0.672 |
| **0.03** | 6 | 588 | 0.646 | 0.708 | 0.876 |
| **0.04** | 2 | 371 | 0.782 | 0.991 | 0.930 |

### 4.9.3 The Stream K-mean Algorithm

The adapted DBSCAN algorithm produces two primary results: the number of clusters, which functions as an input for the Stream k-mean algorithm, and the distribution scheme that recognizes clusters and outliers. If the number of outlier points exceeds 100, it is advisable to utilize a distance function that does not solely depend on Euclidean distance but rather considers the idea that two points do not necessarily have to be reached directly, such as the Canberra distance method. Ultimately, the purpose of the proposed system is to distinguish between humans and bots by applying data stream principles to improve memory usage and execution speed. The ultimate goal of this proposed system is to distinguish between bots and humans by employing principles of a data stream to optimize memory usage and execution speed.

Figure 4.15 illustrates the effectiveness of the Stream K algorithm in identifying potential bot accounts by accurately aggregating Twitter data. Sub-Figures 4.15(a) and 4.15(b) present the clustering results obtained using the Euclidean and Canberra distance metrics, respectively. However, before discussing the results, it is important to understand the matching process used to compare the predicted cluster labels from the clustering algorithm with the original labels.

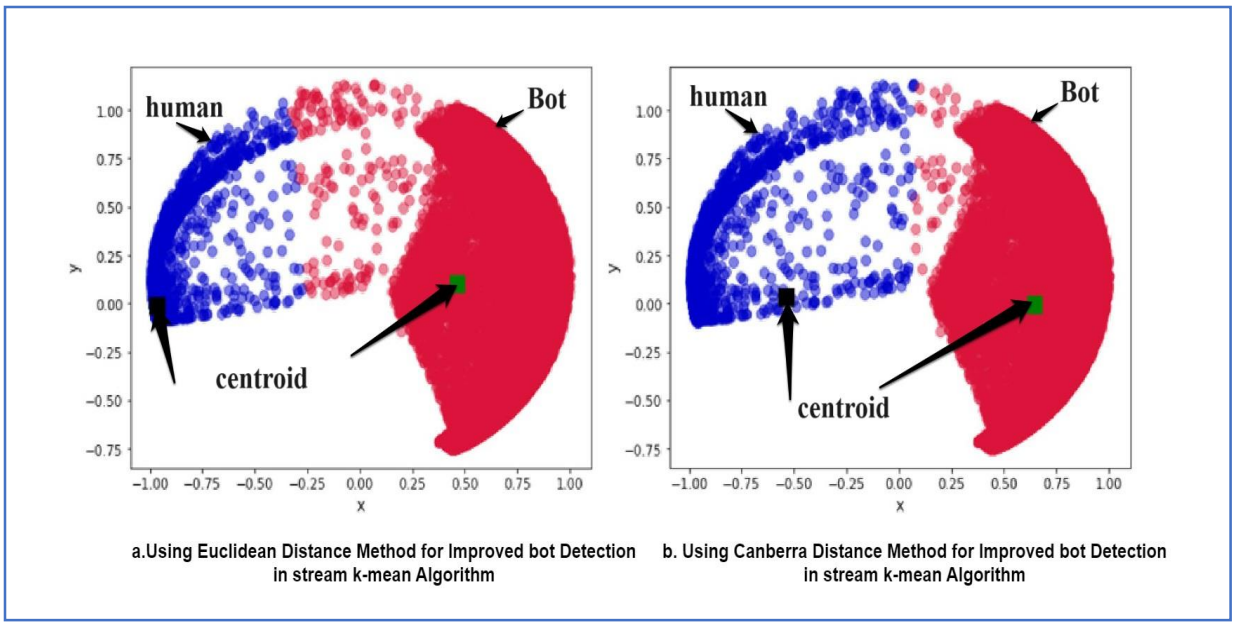
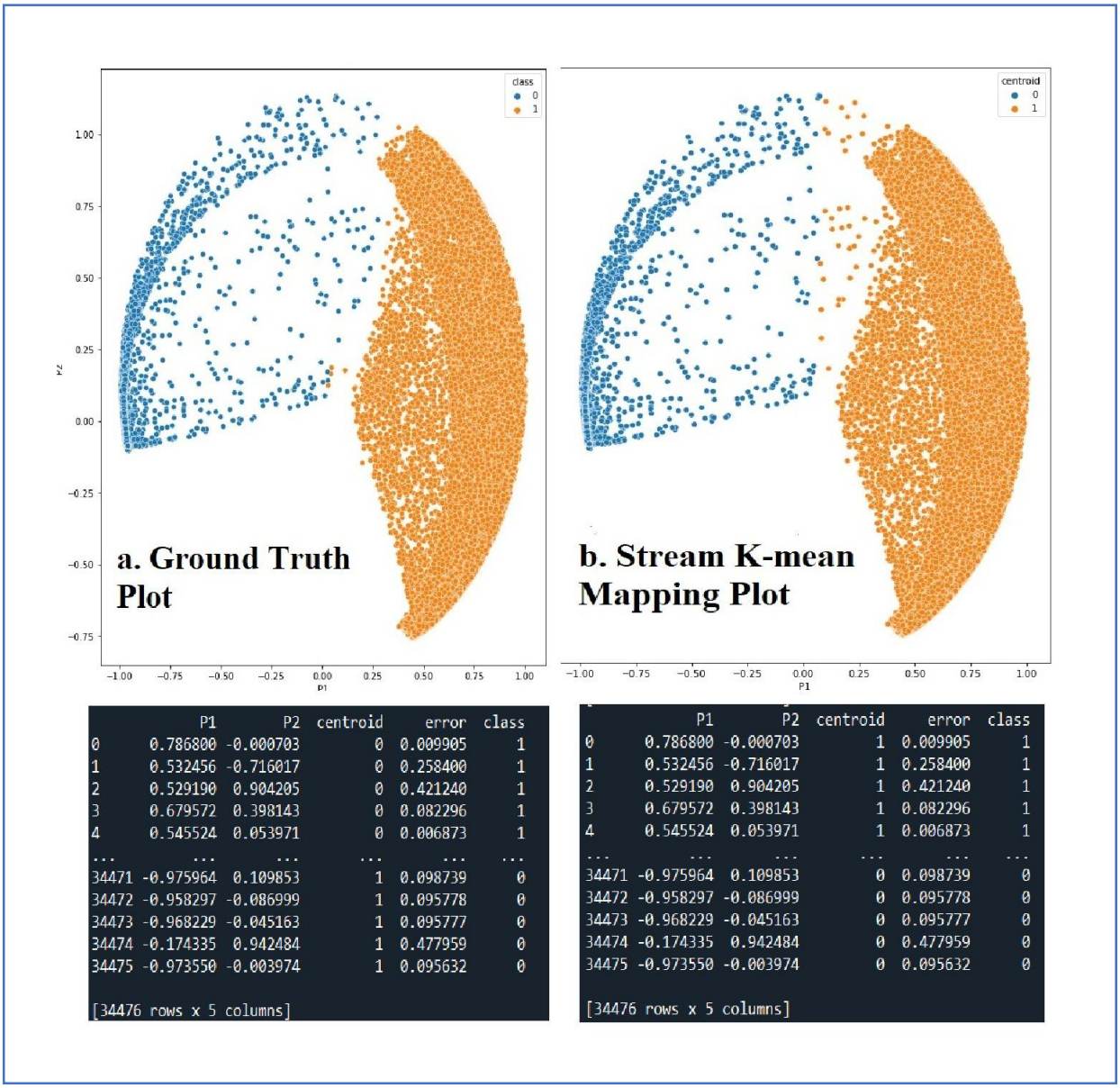



**Figure 4.14: Distinguishing Humans from Bots with Stream K-Mean Algorithm**

This process assesses the quality of the clustering results and evaluates the algorithm's performance as shown in Figure 4.16. The original labels represent the ground truth classes of the data and are assumed to be binary (0 or 1). The clustering algorithm assigns cluster labels to each data point, which may or may not correspond to the original labels. To convert the predicted cluster labels into a format consistent with the original labels, a mapping is applied. In this case, the mapping assigns the value of 1 to predicted label 0 and the value of 0 to predicted label 1. This is done because the original labels are assumed to be either 0 or 1, and the clustering algorithm may have assigned these labels in reverse order.

By applying this mapping, the performance of the clustering algorithm can be evaluated by comparing its predicted labels with the original labels. This can be useful for assessing the quality of the clustering results and comparing the algorithm's performance with other classification methods. The ground truth for the origin dataset labeled is shown in Figure 4.16(a), whereas Figure 4.16(b) displays the clustering algorithm prediction after mapping.




| | P1 | P2 | centroid | error | class |
|---|---|---|---|---|---|
| 0 | 0.786800 | -0.000703 | 0 | 0.009905 | 1 |
| 1 | 0.532456 | -0.716017 | 0 | 0.258400 | 1 |
| 2 | 0.529190 | 0.904205 | 0 | 0.421240 | 1 |
| 3 | 0.679572 | 0.398143 | 0 | 0.082296 | 1 |
| 4 | 0.545524 | 0.053971 | 0 | 0.006873 | 1 |
| ... | ... | ... | ... | ... | ... |
| 34471 | -0.975964 | 0.109853 | 1 | 0.098739 | 0 |
| 34472 | -0.958297 | -0.086999 | 1 | 0.095778 | 0 |
| 34473 | -0.968229 | -0.045163 | 1 | 0.095777 | 0 |
| 34474 | -0.174335 | 0.942484 | 1 | 0.477959 | 0 |
| 34475 | -0.973550 | -0.003974 | 1 | 0.095632 | 0 |

[34476 rows x 5 columns]

| | P1 | P2 | centroid | error | class |
|---|---|---|---|---|---|
| 0 | 0.786800 | -0.000703 | 1 | 0.009905 | 1 |
| 1 | 0.532456 | -0.716017 | 1 | 0.258400 | 1 |
| 2 | 0.529190 | 0.904205 | 1 | 0.421240 | 1 |
| 3 | 0.679572 | 0.398143 | 1 | 0.082296 | 1 |
| 4 | 0.545524 | 0.053971 | 1 | 0.006873 | 1 |
| ... | ... | ... | ... | ... | ... |
| 34471 | -0.975964 | 0.109853 | 0 | 0.098739 | 0 |
| 34472 | -0.958297 | -0.086999 | 0 | 0.095778 | 0 |
| 34473 | -0.968229 | -0.045163 | 0 | 0.095777 | 0 |
| 34474 | -0.174335 | 0.942484 | 0 | 0.477959 | 0 |
| 34475 | -0.973550 | -0.003974 | 0 | 0.095632 | 0 |

[34476 rows x 5 columns]

**Figure 4.15: Mapping origin class with the predicted label of the algorithm**

Notably, in group 0 (indicated by blue in the figure), the Canberra distance outperformed the Euclidean distance. Using the Canberra distance, the total points for group 0 were 14,147, and only 33 points were incorrectly labeled as 1 while using the Euclidean distance, the total points for group 0 were 14,017, and 163 points were incorrectly labeled as 1. Figure 4.17 represents the final outputs of the system, along with the evaluation results, which are shown in Table 4.8.

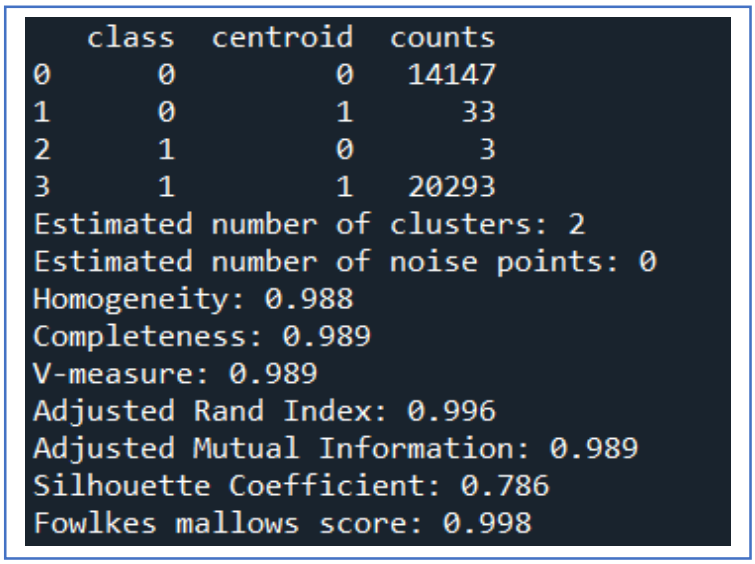



**Figure 4.16: The Final Outputs of The System**

The proposed system's final outputs demonstrate great promise in comparison to previous works, especially by considering the system's ability to automatically distinguish between purchase accounts and bots without requiring any user intervention. This accomplishment highlights the algorithm's efficiency and accuracy and its potential for use in various datasets.

These results demonstrate the importance of selecting an appropriate distance metric when clustering data and highlight the superior performance of the Canberra distance in detecting potential bot accounts in Twitter data.

To calculate the error rate between the numbers 14147 and 14017, the absolute difference between the two numbers should be computed. The absolute difference is simply the positive value of the difference between the two numbers.

Absolute difference = |14147 - 14017| = 130

The error rate can then be calculated by dividing the absolute difference by the actual value of one of the numbers and multiplying by 100 to express it as a percentage.

Error rate = (Absolute difference / Actual value) x 100% (4.2)

Let's choose 14147 as the actual value.

Error rate = (130 / 14147) x 100% = 0.92%

Therefore, the error rate between the numbers 14147 and 14017 is 0.92%.

#### 4.9.3.1 Discussion of the Results

The results of the proposed system provided are quite impressive. It seems that the k-means algorithm was able to cluster the Twitter data into groups with high levels of homogeneity, completeness, and V-measure. This indicates that the algorithm was able to accurately group similar data points in a way that was consistent with human intuition.

In Table 4.8, the high values for the Adjusted Rand Index metric and Adjusted Mutual Information metric suggest that the clusters produced by the algorithm are highly correlated with the true underlying structure of the data. The Silhouette Coefficient of 0.786 is also quite high, indicating that the clusters are well-separated and distinct from one another.

**Table 4.8: Evaluation of "Stream K-means" Using Seven Metrics**

| ID | Metric | Score |
|---|---|---|
| **1** | Homogeneity metric | 0.988 |
| **2** | Completeness metric | 0.989 |
| **3** | V-measure metric | 0.989 |
| **4** | Adjusted Rand Index metric | 0.996 |
| **5** | Adjusted Mutual Information metric | 0.989 |
| **6** | Silhouette Coefficient metric | 0.786 |
| **7** | Fowlkes Mallows metric | 0.998 |

Finally, the Fowlkes Mallows score of 0.998 is an indication of how well the algorithm was able to classify the data points into the appropriate clusters. This score measures the similarity between the true classifications of the data points and the classifications produced by the algorithm.

Overall, it seems that the k-means algorithm was highly effective at clustering the Twitter data and accurately identifying potential bot accounts. The fact that there were zero estimated noise points also suggests that the algorithm was able to effectively separate the signal from the noise in the data. However, it is important to note that the effectiveness of the algorithm may heavily depends on the specific characteristics of the Twitter data being analyzed, as well as the parameters chosen for the k-means algorithm. The superiority of the proposed technique in bot detection using clustering algorithms, particularly stream-based clustering algorithms, is demonstrated in Table 4.9 and compared to several similar studies.

**Table 4.9: Performance Comparison of Bot Detection Methods Based on Stream Clustering**

| Ref | Performance | Method | Details |
|---|---|---|---|
| [200] | F1 = 0.88 | DBScan, K-means++ | Uses a clustering approach to find groups based on features of either tweet account or account usage. |
| [201] | F= 64.1, R =92.9 | Incremental Naïve Bayes-DenStream | The INB-DenStream method categorizes tweets into spam and non-spam clusters, utilizing Naïve Bayes to capture the mean and boundary of microclusters. |
| | F= 63.7, R=92.5 | DenStream | |
| | F=59.6, R=53.4 | StreamKM++ | |
| | F=31.5, R=23.7 | CluStream | |
| [45] | F 1 = 0.87 | HDBSCAN | Employs the HDBSCAN to detect bots by analyzing their previous patterns of retweeting. |

| | | | |
|---|---|---|---|
| [202] | Purity = 0.9 | Uniform Manifold Approximation and Projection, followed by HDBSCAN | A clustering algorithm is utilized to comprehend the features of various kinds of accounts belonging to Twitter's state-sponsored trolls. |
| [203] | Modularity=0.182 | Evolutionary DBSCA | A system designed for additional health monitoring of COVID-19 utilizes the DBSCAN and Louvain method to identify communities in temporal networks. |
| **Proposed method** | Fowlkes Mallows Score=0.998 | DBSCAN+ Stream k-mean | |

## 4.10 Summary

This chapter discusses the proposed system to address Twitter bot detection using unsupervised techniques, particularly clustering algorithms. The dissertation used the Caverlee dataset, which contained over two million tweets from both legitimate users and content polluters. Feature selection was done using the Correlation Attribute Eval (CAE) technique, which ranks features based on their correlation with the target variable.

The effectiveness of the three clustering algorithms (partition, hierarchy, and density) in detecting bots was evaluated using the proposed features. The dissertation implemented four different techniques (k-mean, k-medoid, agglomerative, and DBSCAN) and demonstrated that the proposed features were more effective in detecting bots than the original features.

The dissertation then developed a new integrated approach using three adapted unsupervised algorithms (KNN, DBSCAN, and stream K-mean) to identify Twitter bots. The adaptations made to these algorithms were to address their

prominent problems in clustering and resulted in significantly improved accuracy and faster implementation.

The dissertation demonstrated the effectiveness of the proposed technique in detecting Twitter bots using clustering algorithms, particularly stream-based clustering algorithms. Multiple metrics were used to evaluate the technique, with high values observed for Homogeneity (0.988), Completeness (0.989), V-measure (0.989), Adjusted Rand Index (0.996), Adjusted Mutual Information (0.989), Silhouette Coefficient (0.786), and Fowlkes Mallows Score (0.998).

# CHAPTER FIVE

## CONCLUSIONS AND FUTURE WORKS

## 5.1 Summary and Conclusions

Based on the predefined systematic review conducted throughout this dissertation, the selected studies undergo an in-depth examination to highlight the benefits, challenges, gaps, and recommendations related to bot detection. According to the highlighted challenges in earlier research, the primary aim was to design a comprehensive automated system for detecting bots on Twitter. The first aspect emphasized was the widespread existence of unlabeled Twitter data in real-world scenarios, along with the substantial expenses involved in manually annotating such data. To tackle this challenge, the proposed system for the automatic detection of Twitter bots overcomed the challenge of obtaining labeled data by utilizing unsupervised machine-learning algorithms. The proposed system used three algorithms to achieve auto and adaptive bot detection.

Although researchers have developed powerful models to identify social media bot accounts, bot creators continuously evolve their methods to evade detection. Various bot detection methods were explored that are employed in recent Twitter research and emphasized the selection or extraction of key Twitter features for accurate bot detection.

The main features used in previous literature were also discussed thoroughly, focusing on four criteria: utilized datasets, features, classifiers, and performance measures. By reshaping feature extrapolation techniques, higher standards for Twitter bot detection were established and existing machine-learning applications were utilized by exploring the significant relationships between malicious bots and feature architecture. Descriptive information about Twitter features that interact with the classification of malicious bots was provided in detail.

New features were also created that can improve the accuracy of the clustering algorithm while reducing the dataset's dimensions. By identifying and incorporating relevant features, the overall performance of the clustering algorithms was enhanced.

The third aspect was the integration of three unsupervised algorithms namely, KNN, DBSCAN, and Stream K-means, to identify Twitter bots. The integrated approach was developed based on the strengths and weaknesses of each algorithm. By combining the features of these algorithms, it becomes possible to overcome their limitations and improve the accuracy of bot identification.

Finally, promising results were highlighted by applying the selected features to the adaptive clustering algorithm . The evaluation of the algorithms resulted in an impressive accuracy score of 0.99%. This indicates that the integrated approach effectively identifies Twitter bots and demonstrates the potential of using clustering techniques for bot detection.

In summary, this dissertation concludes that:

1) The integration of multiple unsupervised algorithms offers a comprehensive approach to identifying Twitter bots, and the results demonstrate the effectiveness of this integrated approach.

2) Despite the potential benefits of detecting Twitter malicious bots, this dissertation faces several limitations, including focusing on only one SM network, obtaining or building a sufficient dataset, and selecting an appropriate unsupervised learning method. Overcoming these limitations require careful planning, execution, and continuous monitoring to ensure the proposed system's success in detecting Twitter malicious bots accurately.

## 5.2 Recommendations

Based on analyzing the results, the dissertation reached several recommendations:

1) Exploring unsupervised machine-learning algorithms: Since obtaining labeled data for bot detection can be challenging and expensive, it is recommended to continue exploring and utilizing unsupervised machine-learning algorithms. These algorithms can help automatically detect Twitter bots without the need for labeled data.
2) Key Twitter features: Selecting and extracting key Twitter features can help adapt to the evolving methods employed by bot creators to evade detection.
3) Feature extrapolation techniques: It is also recommended to reshape feature extrapolation techniques to establish higher standards for Twitter bot detection. Investigating the significant relationships between malicious bots and feature architecture can also assist in improving the accuracy of machine-learning applications.
4) Examining previous studies: Conducting a systematic examination of selected studies can lead to highlighting the benefits, challenges, gaps, and recommendations related to bot detection. This can provide a comprehensive understanding of the current state of research and help identify areas for improvement.
5) New features for the clustering algorithm: Extracting new features may help improve the accuracy of the clustering algorithm and reduce dataset dimensions.
6) Cost-effective solutions: Focusing on cost-effective solutions for labeling unlabeled data can be achieved using clustering techniques.

This can help reduce the expenses involved in manual annotation and enable efficient analysis of large-scale Twitter data.

## 5.3 Limitations

While detecting Twitter malicious bots holds potential benefits, this dissertation confronts several limitations:

1) The proposed system only targets the detection of bots on Twitter, disregarding bots in other social media networks or web-based systems. This limitation restricts the system's effectiveness and applicability, as bots can operate across multiple platforms.
2) Acquiring or constructing a comprehensive and reliable dataset of high-quality data represents a research gap. The dataset must be extensive enough to encompass diverse Twitter bot behaviors and provide ample examples for the algorithm's learning process. Additionally, accurate labeling with appropriate ground truth labels is crucial for evaluating the system's performance accurately.
3) Unsupervised learning methods are employed to uncover patterns or groupings in data without prior knowledge of output values. Therefore, selecting an appropriate unsupervised learning method for identifying patterns and classifying Twitter malicious bots presents a challenge.

   Another limitation the of selection of important features that contribute to the detection of bots, particularly in the context of unsupervised machine learning for bot detection. Determining which features are most informative and meaningful can be a complex task due to various factors such as feature redundancy, noise, and the absence of labeled data. Moreover, conventional feature selection techniques like forward or backward selection may not be suitable for unsupervised learning

scenarios, as they rely on the availability of labeled data. This limitation brings an additional level of complexity to the research, as the dissertation must address the selection of an appropriate number of features and explore alternative methods, such as clustering-based approaches or dimensionality reduction techniques, to identify the most relevant features accurately.

## 5.4 Future Work

Here, some suggestions for additional future work are highlighted to enhance the proposed system's performance:

1) Exploring techniques to generate synthetic data points that mimic the characteristics of bot accounts can help expand the training set and improve the model's ability to differentiate between human and bot accounts.
2) Testing the proposed system on multiple datasets, including diverse domains and different social media platforms can evaluate its robustness and generalizability. This can help validate the effectiveness of the system across various contexts and ensure its applicability beyond the specific dataset used in the dissertation.
3) Working on time-series analysis can be achieved by incorporating temporal features and analyzing the posting patterns and activity of Twitter accounts over time. Bots often exhibit distinct behavior patterns, such as posting at regular intervals or engaging in spam-like activities. Time-series analysis can help capture these patterns and contribute to better bot detection.
4) Analyzing the network properties of Twitter accounts such as follower/following relationships and interactions should be performed. Bots often form clusters or exhibit unusual network behavior, which can be

detected through network analysis techniques such as Graph-based algorithms and community detection methods.

5) Including natural language processing techniques to analyze the textual content of tweets and profile descriptions may lead to better findings. Bots may display repetitive or nonsensical language patterns, excessive use of hashtags, or content unrelated to their stated interests. Extracting linguistic features and applying sentiment analysis may provide valuable insights for distinguishing between human and bot accounts.
6) Exploring the potential of analyzing additional metadata associated with tweets, such as location, device information, or posting behavior may reveal new patterns of Bots.
7) Extending the system to perform real-time bot detection can enable an immediate response to bot attacks. This requires optimizing the computational efficiency of the algorithms and developing mechanisms for efficient data streaming and processing.

## References


[1] A. M. Kaplan and M. Haenlein, “Users of the world, unite! The challenges and opportunities of Social Media,” *Bus. Horiz.*, vol. 53, no. 1, pp. 59–68, 2010, doi: 10.1016/j.bushor.2009.09.003.

[2] J. H. Kietzmann, K. Hermkens, I. P. McCarthy, and B. S. Silvestre, “Social media? Get serious! Understanding the functional building blocks of social media,” *Bus. Horiz.*, vol. 54, no. 3, pp. 241–251, 2011, doi: 10.1016/j.bushor.2011.01.005.

[3] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, “The rise of social bots,” *Commun. ACM*, vol. 59, no. 7, pp. 96–104, 2016, doi: 10.1145/2818717.

[4] V. S. Subrahmanian *et al.*, “The DARPA Twitter Bot Challenge,” *Computer (Long. Beach. Calif).*, vol. 49, no. 6, pp. 38–46, 2016, doi: 10.1109/MC.2016.183.

[5] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, “Online Human-Bot Interactions : Detection , Estimation , and Characterization,” no. Icwsm, pp. 280–289, 2017.

[6] S. R. Durugkar, R. Raja, K. K. Nagwanshi, and S. Kumar, *Introduction to data mining*. 2022. doi: 10.1002/9781119792529.ch1.

[7] Z. Ellaky, F. Benabbou, and S. SSOuahabi, “Systematic Literature Review of Social Media Bots Detection Systems,” *J. King Saud Univ. - Comput. Inf. Sci.*, 2023, doi: 10.1016/j.jksuci.2023.04.004.

[8] S. Cresci, F. Lillo, D. Regoli, S. Tardelli, and M. Tesconi, “Cashtag piggybacking: Uncovering spam and bot activity in stock microblogs on

twitter," *ACM Trans. Web*, vol. 13, no. 2, 2019, doi: 10.1145/3313184.

[9] A. H. Wang, "Detecting spam bots in online social networking sites: A machine learning approach," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6166 LNCS, pp. 335–342, 2010, doi: 10.1007/978-3-642-13739-6_25.

[10] J. Ratkiewicz, M. Meiss, M. Conover, B. Gonçalves, A. Flammini, and F. Menczer, "Detecting and Tracking Political Abuse in Social Media," *Proc. Fifth Int. AAAI Conf. Weblogs Soc. Media*, p. 297, 2011.

[11] M. Alrubaian, M. Al-Qurishi, S. M. M. Rahman, and A. Alamri, "A novel prevention mechanism for Sybil attack in online social network," *2015 2nd World Symp. Web Appl. Networking, WSWAN 2015*, no. October, 2015, doi: 10.1109/WSWAN.2015.7210347.

[12] S. Sebastian, S. Ayyappan, and P. Vinod, "Framework for design of Graybot in social network," *Proc. 2014 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2014*, pp. 2331–2336, 2014, doi: 10.1109/ICACCI.2014.6968575.

[13] T. H. V.S. Subrahmanian, Andrew Stevens, Alexander Dekhtyar, Shuyang Gao, Yan Liu), Onur Varol, Prashant Shiralkar, Vinod Vydiswaran, Qiaozhu Mei, "The DARPA TWITTER BOT CHALLENGE," *IEEE Comput. Mag.*, vol. 49, no. 6, pp. 38–46, 2016, doi: 10.1109/MC.2016.183.

[14] N. Abokhodair, D. Yoo, and D. W. McDonald, "Dissecting a social Botnet: Growth, content and influence in twitter," *CSCW 2015 - Proc. 2015 ACM Int. Conf. Comput. Coop. Work Soc. Comput.*, pp. 839–851, 2015, doi: 10.1145/2675133.2675208.

[15] A. Elyashar, M. Fire, D. Kagan, and Y. Elovici, “Guided socialbots: Infiltrating the social networks of specific organizations’ employees,” *AI Commun.*, vol. 29, no. 1, pp. 87–106, 2016, doi: 10.3233/AIC-140650.

[16] O. Goga, G. Venkatadri, and K. P. Gummadi, “The doppelgänger bot attack: Exploring identity impersonation in online social networks,” *Proc. ACM SIGCOMM Internet Meas. Conf. IMC,* vol. 2015-Octob, pp. 141–153, 2015, doi: 10.1145/2815675.2815699.

[17] S. Frenkel, “What Are Spam Bots and Why They’re an Issue in Elon Musk’s Twitter Deal,” *New York Time*, 2022. https://www.nytimes.com/2022/07/09/technology/elon-musk-twitter-spam-bots.html

[18] K.-C. Yang, I. U. Doctoral Student in Informatics, F. Menczer, and I. U. Professor of Informatics and Computer Science, “How many bots are on Twitter? The question is difficult to answer and misses the point,” *the conversation Academic rigour, journalistic flair*, 2022. https://theconversation.com/how-many-bots-are-on-twitter-the-question-is-difficult-to-answer-and-misses-the-point-183425 (accessed Apr. 25, 2023).

[19] T. W. STAFF, “62 percent of all web traffic comes from bots,” *THE WEEK*, 2015. https://theweek.com/articles/454320/62-percent-all-web-traffic-comes-from-bots

[20] S. Dang and S. expands A. chatbot with ability to create Images, “Do spam bots really comprise under 5% of Twitter users? Elon Musk wants to know,” *reuters*, 2022. https://www.reuters.com/technology/do-spam-bots-really-comprise-under-5-twitter-users-elon-musk-wants-know-2022-05-13/ (accessed Apr. 25, 2023).

[21] S. Cresci, A. Spognardi, M. Petrocchi, M. Tesconi, and R. Di Pietro, “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race,” *26th Int. World Wide Web Conf. 2017, WWW 2017 Companion*, pp. 963–972, 2019, doi: 10.1145/3041021.3055135.

[22] M. Aljabri, R. Zagrouba, A. Shaahid, F. Alnasser, A. Saleh, and D. M. Alomari, *Machine learning-based social media bot detection: a comprehensive literature review*, vol. 13, no. 1. Springer Vienna, 2023. doi: 10.1007/s13278-022-01020-5.

[23] V. Garg, T. Koster, and L. J. Camp, “Cross-country analysis of spambots,” *EURASIP J. Inf. Secur.*, vol. 2013, no. 1, 2013, doi: 10.1186/1687-417x-2013-3.

[24] S. Giorgi, L. Ungar, and H. A. Schwartz, “Characterizing Social Spambots by their Human Traits,” *Find. Assoc. Comput. Linguist. ACL-IJCNLP 2021*, pp. 5148–5158, 2021, doi: 10.18653/v1/2021.findings-acl.457.

[25] “What is a social media bot? | Social media bot definition,” *cloudflare*, 2022. https://www.cloudflare.com/learning/bots/what-is-a-social-media-bot/

[26] A. Karataş and S. Şahin, “A review on social bot detection techniques and research directions,” *Proc. Int. Conf. Secur. Cryptol. Conf. Turkey*, no. February, pp. 156–161, 2017, [Online]. Available: https://www.researchgate.net/publication/322853694

[27] A. Aldayel and W. Magdy, “Characterizing the role of bots’ in polarized stance on social media,” *Soc. Netw. Anal. Min.*, vol. 12, no. 1, 2022, doi: 10.1007/s13278-022-00858-z.

[28] D. Murthy *et al.*, “Bots and political influence: A sociotechnical investigation

of social network capital," *Int. J. Commun.*, vol. 10, no. June, pp. 4952–4971, 2016.

[29] E. Alothali, N. Zaki, E. A. Mohamed, and H. Alashwal, "Detecting Social Bots on Twitter: A Literature Review," *Proc. 2018 13th Int. Conf. Innov. Inf. Technol. IIT 2018*, pp. 175–180, 2019, doi: 10.1109/INNOVATIONS.2018.8605995.

[30] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: Human, bot, or cyborg?," *Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC*, pp. 21–30, 2010, doi: 10.1145/1920261.1920265.

[31] Y. Khan, S. Thakur, O. Obiyemi, and E. Adetiba, "Identification of Bots and Cyborgs in the #FeesMustFall Campaign," *Informatics*, vol. 9, no. 1, 2022, doi: 10.3390/informatics9010021.

[32] W. Shahid, Y. Li, D. Staples, G. Amin, S. Hakak, and A. Ghorbani, "Are You a Cyborg, Bot or Human?-A Survey on Detecting Fake News Spreaders," *IEEE Access*, vol. 10, pp. 27069–27083, 2022, doi: 10.1109/ACCESS.2022.3157724.

[33] J. Paavola, T. Helo, H. Jalonen, M. Sartonen, and A. M. Huhtinen, "Understanding the Trolling Phenomenon: The Automated Detection of Bots and Cyborgs in the Social Media," *Eur. Conf. Inf. Warf. Secur. ECCWS*, vol. 2016-Janua, no. 4, pp. 237–244, 2016.

[34] K. Shaukat *et al.*, "Performance comparison and current challenges of using machine learning techniques in cybersecurity," *Energies*, vol. 13, no. 10, 2020, doi: 10.3390/en13102509.

[35] D. M. G. De Morais and L. A. Digiampietri, "Methods and Challenges in

Social Bots Detection: A Systematic Review," *ACM Int. Conf. Proceeding Ser.*, pp. 21–28, 2021, doi: 10.1145/3466933.3466973.

[36] Frederik Bussler, "Bots Are Taking Over the Web—Here's How to Fight Back," *fingerprint.com*, 2022. https://fingerprint.com/blog/what-are-bots-how-to-detect-bots/ (accessed Apr. 15, 2023).

[37] S. T. K. Jan *et al.*, "Throwing darts in the dark? detecting bots with limited data using neural data augmentation," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2020-May, pp. 1190–1206, 2020, doi: 10.1109/SP40000.2020.00079.

[38] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," *SIGIR 2010 Proc. - 33rd Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, no. June, pp. 435–442, 2010, doi: 10.1145/1835449.1835522.

[39] E. Elakkiya, S. Selvakumar, and R. L. Velusamy, "CIFAS: Community Inspired Firefly Algorithm with fuzzy cross-entropy for feature selection in Twitter Spam detection," *2020 11th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2020*, 2020, doi: 10.1109/ICCCNT49239.2020.9225321.

[40] K. C. Yang, O. Varol, C. A. Davis, E. Ferrara, A. Flammini, and F. Menczer, "Arming the public with artificial intelligence to counter social bots," *Hum. Behav. Emerg. Technol.*, vol. 1, no. 1, pp. 48–61, 2019, doi: 10.1002/hbe2.115.

[41] H. Shukla, N. Jagtap, and B. Patil, "Enhanced Twitter bot detection using ensemble machine learning," *Proc. 6th Int. Conf. Inven. Comput. Technol. ICICT 2021*, pp. 930–936, 2021, doi: 10.1109/ICICT50816.2021.9358734.

[42] A. Anwar and U. Yaqub, "Bot detection in twitter landscape using

unsupervised learning," *ACM Int. Conf. Proceeding Ser.*, no. June, pp. 329–330, 2020, doi: 10.1145/3396956.3401801.

[43] B. Wu, L. Liu, Y. Yang, K. Zheng, and X. Wang, "Using improved conditional generative adversarial networks to detect social bots on Twitter," *IEEE Access*, vol. 8, pp. 36664–36680, 2020, doi: 10.1109/ACCESS.2020.2975630.

[44] N. Chavoshi, H. Hamooni, and A. Mueen, "DeBot: Twitter Bot Detection via Warped Correlation," no. October 2017, pp. 817–822, 2017, doi: 10.1109/icdm.2016.0096.

[45] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "RTbust: Exploiting temporal patterns for botnet detection on twitter," *WebSci 2019 - Proc. 11th ACM Conf. Web Sci.*, pp. 183–192, 2019, doi: 10.1145/3292522.3326015.

[46] L. Wu, N. A. Dodoo, T. J. Wen, and L. Ke, "Understanding Twitter conversations about artificial intelligence in advertising based on natural language processing," *Int. J. Advert.*, vol. 41, no. 4, pp. 685–702, 2022, doi: 10.1080/02650487.2021.1920218.

[47] P. Shi, Z. Zhang, and K. K. R. Choo, "Detecting Malicious Social Bots Based on Clickstream Sequences," *IEEE Access*, vol. 7, pp. 28855–28862, 2019, doi: 10.1109/ACCESS.2019.2901864.

[48] A. Dorri, M. Abadi, and M. Dadfarnia, "SocialBotHunter: Botnet detection in twitter-like social networking services using semi-supervised collective classification," *Proc. - IEEE 16th Int. Conf. Dependable, Auton. Secur. Comput. IEEE 16th Int. Conf. Pervasive Intell. Comput. IEEE 4th Int. Conf. Big Data Intell. Comput. IEEE 3*, no. December 2021, pp. 496–503, 2018,

doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00097.

[49] S. A. NA Ibupoto, Z Rasheed, “Sentiment Analysis On Current Political Topics In Pakistan ’ s Twitter User Bases,” *Res. Sq.*, vol. v1, no. This work is licensed under a Creative Commons Attribution 4.0 International License, pp. 1–14, 2022, doi: https://doi.org/10.21203/rs.3.rs-2095172/v1 License:

[50] Y. Jiang, Z. Li, and X. Ye, “Understanding demographic and socioeconomic biases of geotagged Twitter users at the county level,” *Cartogr. Geogr. Inf. Sci.*, vol. 46, no. 3, pp. 228–242, 2019, doi: 10.1080/15230406.2018.1434834.

[51] M. P. Nirali Patela, “Survey of Feature-based Bot Detection Methodologies,” *Open J. Math. Phys.*, vol. Volume-I, no. 1, pp. 21–24, 2020, doi: 10.21428/c53615e7.6d3c0080.

[52] B. Y. E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, “The rise of social bots,” *Commun. ACM*, vol. 59, pp. 96–104, 2016, doi: doi: 10.1145/2818717.

[53] Z. Yang, X. Chen, H. Wang, W. Wang, Z. Miao, and T. Jiang, “A New Joint Approach with Temporal and Profile Information for Social Bot Detection,” *Secur. Commun. Networks*, vol. 2022, 2022, doi: 10.1155/2022/9119388.

[54] J. Brier and lia dwi jayanti, *Introduction to Machine Learning*, Third Edit., vol. 21, no. 1. Massachusetts Institute of Technology All, 2020. [Online]. Available: http://journal.um-surabaya.ac.id/index.php/JKM/article/view/2203

[55] A. M. Araujo, A. Bergamini de Neira, and M. Nogueira, “Autonomous machine learning for early bot detection in the internet of things,” *Digit. Commun. Networks*, 2022, doi: 10.1016/j.dcan.2022.05.011.

[56] M. Orabi, D. Mouheb, Z. Al Aghbari, and I. Kamel, “Detection of Bots in Social Media: A Systematic Review,” *Inf. Process. Manag.*, vol. 57, no. 4, p. 102250, 2020, doi: 10.1016/j.ipm.2020.102250.

[57] M. HARGRAVE, F. Bio, LinkedIn, Twitter, as well as analyzing and valuing companies. Marshall Hargrave is a stock analyst and writer with 10+ years of experience covering stocks and markets, and L. about our editorial Policies, “Crowdsourcing: Definition, How It Works, Types, and Examples,” *Investopedia is a part of the Dotdash Meredith publishing family.*, 2022. https://www.investopedia.com/terms/c/crowdsourcing.asp

[58] G. Wang *et al.*, “Social Turing Tests: Crowdsourcing Sybil Detection Mobile Application Security View project Celltower Localization View project Social Turing Tests: Crowdsourcing Sybil Detection,” 2012, [Online]. Available: https://www.researchgate.net/publication/224973061

[59] A. Alarifi, M. Alsaleh, and A. M. Al-Salman, “Twitter turing test: Identifying social machines,” *Inf. Sci. (Ny).*, vol. 372, pp. 332–346, 2016, doi: 10.1016/j.ins.2016.08.036.

[60] M. Kouvela, I. Dimitriadis, and A. Vakali, “Bot-Detective: An explainable Twitter bot detection service with crowdsourcing functionalities,” *Proc. 12th Int. Conf. Manag. Digit. Ecosyst. MEDES 2020*, pp. 55–63, 2020, doi: 10.1145/3415958.3433075.

[61] L. A. Cornelissen, R. J. Barnett, P. Schoonwinkel, B. D. Eichstadt, and H. B. Magodla, “A network topology approach to bot classification,” *ACM Int. Conf. Proceeding Ser.*, pp. 79–88, 2018, doi: 10.1145/3278681.3278692.

[62] S. Hurtado, P. Ray, and R. Marculescu, “Bot detection in reddit political discussion,” *Soc. 2019 - Proc. 2019 4th Int. Work. Soc. Sens.*, pp. 30–35,

2019, doi: 10.1145/3313294.3313386.

[63] N. Abu-El-Rub and A. Mueen, “BotCamp: Bot-driven interactions in social campaigns,” *Web Conf. 2019 - Proc. World Wide Web Conf. WWW 2019*, pp. 2529–2535, 2019, doi: 10.1145/3308558.3313420.

[64] L. Alkulaib, L. Zhang, Y. Sun, and C.-T. Lu, “Twitter Bot Identification: An Anomaly Detection Approach,” pp. 3577–3585, 2023, doi: 10.1109/bigdata55660.2022.10020919.

[65] J. Echeverria and S. Zhou, “Discovery, retrieval, and analysis of the’Star wars’ botnet in twitter,” *Proc. 2017 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Mining, ASONAM 2017*, pp. 1–8, 2017, doi: 10.1145/3110025.3110074.

[66] “What is Machine Learning?,” *IBM*. https://www.ibm.com/topics/machine-learning (accessed Mar. 28, 2023).

[67] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[68] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*, The 3rd ed., vol. 9781107015. Stanford University, 2014. doi: 10.1017/CBO9781139058452.

[69] Ryan, Cooper, and Tauer, *Data Mining-Concepts and Techniques*, Third Edit. 225Wyman Street,Waltham, MA 02451, USA: 2012 by Elsevier Inc. All rights reserved, 2013.

[70] A. Dave, “An Introduction to Unsupervised Learning for Trading,” *QuantInsti Quantitative Learning Pvt. Ltd.*, 2021.

https://blog.quantinsti.com/unsupervised-learning/ (accessed Mar. 30, 2023).

[71] F. Informa, W. R. Number, M. House, M. Street, and J. C. Dunn, "Well-Separated Clusters and Optimal Fuzzy Partitions Well-Separated Clusters and Optimal Fuzzy Partitions," no. June 2013, pp. 37–41, 2008.

[72] T. Chadjipadelis *et al.*, *Studies in Classification, Data Analysis, and Knowledge Organization Data Analysis and Rationality in a Complex World*. Switzerland: Springer Nature Switzerland AG 2021, 2021. [Online]. Available: http://www.springer.com/series/1564

[73] L. Lorenzo and J. Arroyo, *Online risk-based portfolio allocation on subsets of crypto assets applying a prototype-based clustering algorithm*, vol. 9, no. 1. Springer Berlin Heidelberg, 2023. doi: 10.1186/s40854-022-00438-2.

[74] L. Lorenzo and J. Arroyo, "Analysis of the cryptocurrency market using different prototype-based clustering techniques," *Financ. Innov.*, vol. 8, no. 1, 2022, doi: 10.1186/s40854-021-00310-9.

[75] VijayPrakash, "Machine_Learning," *GitHub, Inc.*, 2020. https://github.com/VijayPrakashReddy-k/Machine_Learning (accessed Apr. 01, 2023).

[76] Y. Jiang, X. Cui, M. Bennis, F. C. Zheng, B. Fan, and X. You, "Cooperative caching in fog radio access networks: A graph-based approach," *IET Commun.*, vol. 13, no. 20, pp. 3519–3528, 2019, doi: 10.1049/iet-com.2019.0436.

[77] B. Hufnagl and H. Lohninger, "A graph-based clustering method with special focus on hyperspectral imaging," *Anal. Chim. Acta*, vol. 1097, no. xxxx, pp. 37–48, 2020, doi: 10.1016/j.aca.2019.10.071.

[78] L. T. Li, Z. Y. Xiong, Q. Z. Dai, Y. F. Zha, Y. F. Zhang, and J. P. Dan, “A novel graph-based clustering method using noise cutting,” *Inf. Syst.*, vol. 91, p. 101504, 2020, doi: 10.1016/j.is.2020.101504.

[79] R. J. G. B. Campello, P. Kröger, J. Sander, and A. Zimek, “Density-based clustering,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 10, no. 2, 2020, doi: 10.1002/widm.1343.

[80] R. Devika, S. Revathy, U. Sai Surriya Priyanka, and V. Subramaniya Swamy, “Survey on clustering techniques in Twitter data,” *Proc. 2nd Int. Conf. Comput. Methodol. Commun. ICCMC 2018*, vol. 5, no. 2, pp. 1073–1077, 2018, doi: 10.1109/ICCMC.2018.8487969.

[81] S. Pandit and S. Gupta, “A Comparative Study on Distance Measuring Approaches for Clustering,” *Int. J. Res. Comput. Sci.*, vol. 2, no. 1, pp. 29–31, 2011, doi: 10.7815/ijorcs.21.2011.011.

[82] A. Vimal, S. R. Valluri, and K. Karlapalem, “An Experiment with Distance Measures for Clustering ∗,” *Matrix*, no. October, pp. 5–8, 2008.

[83] H. Xie *et al.*, “Improving K-means clustering with enhanced Firefly Algorithms,” *Appl. Soft Comput. J.*, vol. 84, p. 105763, 2019, doi: 10.1016/j.asoc.2019.105763.

[84] S.-H. Cha, “Comprehensive survey on distance/similarity measures between probability density functions,” *City*, vol. 1, no. 2, p. 1, 2007.

[85] Y. Xu and P. Wang, “An enhanced squared exponential kernel with manhattan similarity measure for high dimensional Gaussian process models,” *Proc. ASME Des. Eng. Tech. Conf.*, vol. 3B-2021, pp. 1–10, 2021, doi: 10.1115/DETC2021-71445.

[86] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics),* vol. 1973, pp. 420–434, 2001, doi: 10.1007/3-540-44503-x_27.

[87] M. S. Rao, B. E. Reddy, K. Ramana, K. Prasanna, and S. Singh, “Texture classification using Minkowski distance measure-based clustering for feature selection,” *J. Electron. Imaging*, vol. 31, no. 04, pp. 1–20, 2021, doi: 10.1117/1.jei.31.4.041204.

[88] A. A. Thant and S. M. Aye, “Euclidean, Manhattan and Minkowski Distance Methods For Clustering Algorithms,” *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 7, no. 3, pp. 553–559, 2020, doi: 10.32628/ijsrset2073118.

[89] D. M. SAPUTRA, D. SAPUTRA, and L. D. OSWARI, “Effect of Distance Metrics in Determining K-Value in K-Means Clustering Using Elbow and Silhouette Method,” vol. 172, no. Siconian 2019, pp. 341–346, 2020, doi: 10.2991/aisr.k.200424.051.

[90] Mahmoud Harmouch, “17 types of similarity and dissimilarity measures used in data science. | by Mahmoud Harmouch | Towards Data Science,” *Medium/Published in Towards Data Science*. https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681 (accessed Apr. 02, 2023).

[91] M. Faisal, E. M. Zamzami, and Sutarman, “Comparative Analysis of Inter-Centroid K-Means Performance using Euclidean Distance, Canberra Distance and Manhattan Distance,” *J. Phys. Conf. Ser.*, vol. 1566, no. 1, 2020, doi: 10.1088/1742-6596/1566/1/012112.

[92] H. Ren, Y. Gao, and T. Yang, “A Novel Regret Theory-Based Decision-

Making Method Combined with the Intuitionistic Fuzzy Canberra Distance," *Discret. Dyn. Nat. Soc.*, vol. 2020, 2020, doi: 10.1155/2020/8848031.

[93] M. Santosh and A. Sharma, "Proposed framework for emotion recognition using canberra distance classifier," *J. Comput. Theor. Nanosci.*, vol. 16, no. 9, pp. 3778–3782, 2019, doi: 10.1166/jctn.2019.8250.

[94] X. Gao and M. Yang, "Understanding and enhancement of internal clustering validation indexes for categorical data," *Algorithms*, vol. 11, no. 11, 2018, doi: 10.3390/a11110177.

[95] M. Eminagaoglu, "A new similarity measure for vector space models in text classification and information retrieval," *J. Inf. Sci.*, vol. 48, no. 4, pp. 463–476, 2022, doi: 10.1177/0165551520968055.

[96] D. W. Roberts, "Distance, dissimilarity, and mean–variance ratios in ordination," *Methods Ecol. Evol.*, vol. 8, no. 11, pp. 1398–1407, 2017, doi: 10.1111/2041-210X.12739.

[97] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," *Proc. Annu. ACM-SIAM Symp. Discret. Algorithms*, vol. 07-09-Janu, pp. 1027–1035, 2007.

[98] A. Deshpande and I. I. T. Mandi, "Robust k -means ++," vol. 124, 2020.

[99] T. Su and J. G. Dy, "In search of deterministic methods for initializing K-means and Gaussian mixture clustering," *Intell. Data Anal.*, vol. 11, no. 4, pp. 319–338, 2007, doi: 10.3233/ida-2007-11402.

[100] M. E. Celebi and H. A. Kingravi, "Deterministic initialization of the K-means algorithm using hierarchical clustering," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 26, no. 7, 2012, doi: 10.1142/S0218001412500188.

[101] L. Muflikhah, Widodo, W. F. Mahmudy, and Solimun, "DNA Sequence of Hepatitis B Virus Clustering Using Hierarchical k-Means Algorithm," *ICETAS 2019 - 2019 6th IEEE Int. Conf. Eng. Technol. Appl. Sci.*, pp. 19–22, 2019, doi: 10.1109/ICETAS48360.2019.9117565.

[102] J. Shen, S. I. Chang, E. S. Lee, Y. Deng, and S. J. Brown, "Determination of cluster number in clustering microarray data," *Appl. Math. Comput.*, vol. 169, no. 2, pp. 1172–1185, 2005, doi: 10.1016/j.amc.2004.10.076.

[103] A. Et-Taleby, M. Boussetta, and M. Benslimane, "Faults detection for photovoltaic field based on k-means, elbow, and average silhouette techniques through the segmentation of a thermal image," *Int. J. Photoenergy*, vol. 2020, 2020, doi: 10.1155/2020/6617597.

[104] E. Umargono, J. E. Suseno, and V. G. S. K., "K-Means Clustering Optimization using the Elbow Method and Early Centroid Determination Based-on Mean and Median," vol. 474, no. Isstec 2019, pp. 234–240, 2020, doi: 10.5220/0009908402340240.

[105] M. Cui, "Introduction to the K-Means Clustering Algorithm Based on the Elbow Method," *Clausius Sci. Press*, vol. 3, pp. 5–8, 2020, doi: 10.23977/accaf.2020.010102.

[106] R. Nainggolan, R. Perangin-Angin, E. Simarmata, and A. F. Tarigan, "Improved the Performance of the K-Means Cluster Using the Sum of Squared Error (SSE) optimized by using the Elbow Method," *J. Phys. Conf. Ser.*, vol. 1361, no. 1, 2019, doi: 10.1088/1742-6596/1361/1/012015.

[107] P. Dangeti, "The elbow method," *O'Reilly*. https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/c71ea970-0f3c-4973-8d3a-b09a7a6553c1.xhtml

[108] E. Schubert, “Stop using the elbow criterion for k-means and how to choose the number of clusters instead,” 2022, [Online]. Available: http://arxiv.org/abs/2212.12189

[109] R. Sammouda and A. El-Zaart, “An Optimized Approach for Prostate Image Segmentation Using K-Means Clustering Algorithm with Elbow Method,” *Comput. Intell. Neurosci.*, vol. 2021, 2021, doi: 10.1155/2021/4553832.

[110] A. Naghizadeh and D. N. Metaxas, “Condensed silhouette: An optimized filtering process for cluster selection in K-means,” *Procedia Comput. Sci.*, vol. 176, pp. 205–214, 2020, doi: 10.1016/j.procs.2020.08.022.

[111] H. B. Tambunan, D. H. Barus, J. Hartono, A. S. Alam, D. A. Nugraha, and H. H. H. Usman, “Electrical peak load clustering analysis using K-means algorithm and silhouette coefficient,” *Proceeding - 2nd Int. Conf. Technol. Policy Electr. Power Energy, ICT-PEP 2020*, pp. 258–262, 2020, doi: 10.1109/ICT-PEP50916.2020.9249773.

[112] H. Humaira and R. Rasyidah, “Determining The Appropiate Cluster Number Using Elbow Method for K-Means Algorithm,” 2020, doi: 10.4108/eai.24-1-2018.2292388.

[113] C. Yuan and H. Yang, “Research on K-Value Selection Method of K-Means Clustering Algorithm,” *J*, vol. 2, no. 2, pp. 226–235, 2019, doi: 10.3390/j2020016.

[114] Z. Li, Y. Li, W. Lu, and J. Huang, “Crowdsourcing Logistics Pricing Optimization Model Based on DBSCAN Clustering Algorithm,” *IEEE Access*, vol. 8, pp. 92615–92626, 2020, doi: 10.1109/ACCESS.2020.2995063.

[115] Pallavi Pandey, “DBSCAN Clustering,” *machine learning geek*. https://machinelearninggeek.com/dbscan-clustering/ (accessed Apr. 06, 2023).

[116] R. He *et al.*, “Clustering Enabled Wireless Channel Modeling Using Big Data Algorithms,” *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 177–183, 2018, doi: 10.1109/MCOM.2018.1700701.

[117] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *Proc. 2nd Int. Conf. Knowl. Discov. Data Min.*, pp. 226–231, 1996.

[118] J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications,” *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, 1998, doi: 10.1023/A:1009745219419.

[119] N. Valarmathy and S. Krishnaveni, “WITHDRAWN: A novel method to enhance the performance evaluation of DBSCAN clustering algorithm using different distinguished metrics,” *Mater. Today Proc.*, no. xxxx, 2020, doi: 10.1016/j.matpr.2020.09.623.

[120] Q. Zhu, X. Tang, and Z. Liu, “Revised DBSCAN Clustering Algorithm Based on Dual Grid,” *Proc. 32nd Chinese Control Decis. Conf. CCDC 2020*, pp. 3461–3466, 2020, doi: 10.1109/CCDC49329.2020.9163926.

[121] V. Lytvynenko, I. Lurie, J. Krejci, M. Voronenko, N. Savina, and M. A. Taif, “Two step density-based object-inductive clustering algorithm,” *CEUR Workshop Proc.*, vol. 2386, pp. 117–135, 2019.

[122] S. Shriram and E. Sivasankar, “Anomaly Detection on Shuttle data using

Unsupervised Learning Techniques," *Proc. 2019 Int. Conf. Comput. Intell. Knowl. Econ. ICCIKE 2019*, pp. 221–225, 2019, doi: 10.1109/ICCIKE47802.2019.9004325.

[123] X. Zheng *et al.*, "Non-Parametric Unsupervised Domain Adaptation for Neural Machine Translation," *Find. Assoc. Comput. Linguist. Find. ACL EMNLP 2021*, pp. 4234–4241, 2021, doi: 10.18653/v1/2021.findings-emnlp.358.

[124] D. Tarlow, K. Swersky, L. Charlin, I. Sutskever, and R. S. Zemel, "Stochastic k-neighborhood selection for supervised and unsupervised learning," *30th Int. Conf. Mach. Learn. ICML 2013*, vol. 28, no. PART 2, pp. 1236–1244, 2013.

[125] R. Zhao, W. Ouyang, and X. Wang, "Unsupervised salience learning for person re-identification," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3586–3593, 2013, doi: 10.1109/CVPR.2013.460.

[126] J.-C. CHOUINARD, "k-Nearest Neighbors (KNN) in Python - JC Chouinard," *JC CHOUINARD*, 2022. https://www.jcchouinard.com/k-nearest-neighbors/ (accessed Apr. 08, 2023).

[127] S. Li and N. Amenta, "Brute-Force k -Nearest Neighbors Search on the GPU," in *International Conference on Similarity Search and Applications*, 2015, vol. 9371, pp. 259–270. doi: https://doi.org/10.1007/978-3-319-25087-8_25.

[128] F. Magliani, K. McGuinness, E. Mohedano, and A. Prati, "An efficient approximate kNN graph method for diffusion on image retrieval," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11752 LNCS, pp. 537–548, 2019, doi: 10.1007/978-3-

030-30645-8_49.

[129] J. Adamczyk, “k nearest neighbors computational complexity,” *Published in Towards Data Science*. https://towardsdatascience.com/k-nearest-neighbors-computational-complexity-502d2c440d5 (accessed Apr. 08, 2023).

[130] H. M. Kakde, “Range Searching using Kd Tree,” 2005. [Online]. Available: http://www.cs.utah.edu/~lifeifei/cis5930/kdtree.pdf

[131] R. Pinkham, S. Zeng, and Z. Zhang, “QuickNN: Memory and Performance Optimization of k-d Tree Based Nearest Neighbor Search for 3D Point Clouds,” *Proc. - 2020 IEEE Int. Symp. High Perform. Comput. Archit. HPCA 2020*, pp. 180–192, 2020, doi: 10.1109/HPCA47549.2020.00024.

[132] A. N. Member, J. B. Member, S. Karumanchi, P. Tavallali, and B. Kennedy, “Deterministic Iteratively Built KD-Tree with KNN Search for Exact Applications ( September 2020 ),” no. September, pp. 1–7, 2020.

[133] P. Anagnostou, P. Barbas, A. G. Vrahatis, and S. K. Tasoulis, “Approximate kNN Classification for Biomedical Data,” *Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020*, vol. i, pp. 3602–3607, 2020, doi: 10.1109/BigData50022.2020.9378126.

[134] S. M. Omohundro, “Five Balltree Construction Algorithms,” *Int. Comput. Sci. Inst. Berkeley*, pp. 1–22, 1989, [Online]. Available: https://scholar.googleusercontent.com/scholar?q=cache:sog078YXsbsJ:scholar.google.com/+Stephen+M+Omohundro,+1989&hl=en&as_sdt=0,5

[135] Y. Pan, Z. Pan, Y. Wang, and W. Wang, “A new fast search algorithm for exact k-nearest neighbors based on optimal triangle-inequality-based check strategy,” *Knowledge-Based Syst.*, vol. 189, p. 105088, 2020, doi:

10.1016/j.knosys.2019.105088.

[136] R. Rahim and A. S. Ahmar, “Cross-Validation and Validation Set Methods for Choosing K in KNN Algorithm for Healthcare Case Study,” *JINAV J. Inf. Vis.*, vol. 3, no. 1, pp. 57–61, 2022, doi: 10.35877/454ri.jinav1557.

[137] Y. Zhongguo, L. Hongqi, Z. Liping, L. Qiang, and S. Ali, “A case based method to predict optimal k value for k-NN algorithm,” *J. Intell. Fuzzy Syst.*, vol. 33, no. 1, pp. 55–65, 2017, doi: 10.3233/JIFS-161062.

[138] Z. Pan, Y. Wang, and Y. Pan, “A new locally adaptive k-nearest neighbor algorithm based on discrimination class,” *Knowledge-Based Syst.*, vol. 204, p. 106185, 2020, doi: 10.1016/j.knosys.2020.106185.

[139] U. Lall and A. Sharma, “A nearest neighbor bootstrap for resampling hydrologic time series,” *Water Resour. Res.*, vol. 32, no. 3, pp. 679–693, 1996, doi: 10.1029/95WR02966.

[140] H. Liu, S. Zhang, J. Zhao, X. Zhao, and Y. Mo, “A new classification algorithm using mutual nearest neighbors,” *Proc. - 9th Int. Conf. Grid Cloud Comput. GCC 2010*, pp. 52–57, 2010, doi: 10.1109/GCC.2010.23.

[141] F. Bulut and M. F. Amasyali, “Locally adaptive k parameter selection for nearest neighbor classifier: one nearest cluster,” *Pattern Anal. Appl.*, vol. 20, no. 2, pp. 415–425, 2017, doi: 10.1007/s10044-015-0504-0.

[142] G. Bhattacharya, K. Ghosh, and A. S. Chowdhury, “Test point specific k estimation for kNN classifier,” *Proc. - Int. Conf. Pattern Recognit.*, pp. 1478–1483, 2014, doi: 10.1109/ICPR.2014.263.

[143] A. B. Hassanat, M. A. Abbadi, G. A. Altarawneh, and A. A. Alhasanat, “Solving the Problem of the K Parameter in the KNN Classifier Using an

Ensemble Learning Approach," vol. 12, no. 8, pp. 33–39, 2014, [Online]. Available: http://arxiv.org/abs/1409.0919

[144] M. T. Ö. (eds. . Christian S. Jensen, Richard T. Snodgrass (auth.), LING LIU, *Encyclopedia of Database Systems*, Volum-1. Atlant,/USA: Springer US, 2009. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_133

[145] S. Karanam, "Curse of Dimensionality — A 'Curse' to Machine Learning," *Towards Data Science*, 2021. https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb (accessed Apr. 08, 2023).

[146] B. D.-M. S. of B. Shetty, "What Is the Curse of Dimensionality?," *Built In is the online community for startups and tech companies. Find startup jobs, tech news and events.*, 2022. https://builtin.com/data-science/curse-dimensionality (accessed Apr. 08, 2023).

[147] D. M. Beskow and K. M. Carley, "Its all in a name: detecting and labeling bots by their name," *Comput. Math. Organ. Theory*, vol. 25, no. 1, pp. 24–35, 2019, doi: 10.1007/s10588-018-09290-1.

[148] Z. Manbari, F. AkhlaghianTab, and C. Salavati, "Hybrid fast unsupervised feature selection for high-dimensional data," *Expert Syst. Appl.*, vol. 124, no. June, pp. 97–118, 2019, doi: 10.1016/j.eswa.2019.01.016.

[149] T. Platform, "Tweet metadata timeline | Docs | Twitter Developer Platform," *Twitter, Inc.* https://developer.twitter.com/en/docs/twitter-api/enterprise/data-dictionary/tweet-timeline (accessed Apr. 10, 2023).

[150] A. K. Ojo, “Improved Model for Detecting Fake Profiles in Online Social Network: A Case Study of Twitter,” *J. Adv. Math. Comput. Sci.*, vol. 33, no. 4, pp. 1–17, 2019, doi: 10.9734/jamcs/2019/v33i430187.

[151] N. S. Murugan and G. U. Devi, “Feature extraction using LR-PCA hybridization on twitter data and classification accuracy using machine learning algorithms,” *Cluster Comput.*, vol. 22, pp. 13965–13974, 2019, doi: 10.1007/s10586-018-2158-3.

[152] M. S. Al-Rakhami and A. M. Al-Amri, “Lies Kill, Facts Save: Detecting COVID-19 Misinformation in Twitter,” *IEEE Access*, vol. 8, pp. 155961–155970, 2020, doi: 10.1109/ACCESS.2020.3019600.

[153] A. Balestrucci, R. De Nicola, M. Petrocchi, and C. Trubiani, “A behavioural analysis of credulous Twitter users,” *Online Soc. Networks Media*, vol. 23, no. January, p. 100133, 2021, doi: 10.1016/j.osnem.2021.100133.

[154] R. R. Rostami and S. Karbasi, “Detecting fake accounts on twitter social network using multi-objective hybrid feature selection approach,” *Webology*, vol. 17, no. 1, pp. 1–18, 2020, doi: 10.14704/WEB/V17I1/A204.

[155] N. Patel and M. Panchal, “Survey of Feature-based Bot Detection Methodologies,” vol. 2020, pp. 1–4, 2020.

[156] P. G. Pratama and N. A. Rakhmawati, “Social bot detection on 2019 Indonesia president candidate’s supporter’s tweets,” *Procedia Comput. Sci.*, vol. 161, pp. 813–820, 2019, doi: 10.1016/j.procs.2019.11.187.

[157] M. B. Abdulrazzaq and J. N. Saeed, “A Comparison of Three Classification Algorithms for Handwritten Digit Recognition,” *2019 Int. Conf. Adv. Sci. Eng. ICOASE 2019*, no. 46080, pp. 58–63, 2019, doi:

10.1109/ICOASE.2019.8723702.

[158] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on Twitter," *Comput. Secur.*, vol. 91, 2020, doi: 10.1016/j.cose.2020.101715.

[159] A. Rasool, R. Tao, M. Kamyab, and S. Hayat, "GAWA-A feature selection method for hybrid sentiment classification," *IEEE Access*, vol. 8, pp. 191850–191861, 2020, doi: 10.1109/ACCESS.2020.3030642.

[160] H. Rehioui and A. Idrissi, "New clustering algorithms for twitter sentiment analysis," *IEEE Syst. J.*, vol. 14, no. 1, pp. 530–537, 2020, doi: 10.1109/JSYST.2019.2912759.

[161] N. K. Suchetha, A. Nikhil, and P. Hrudya, "Comparing the wrapper feature selection evaluators on twitter sentiment classification," *ICCIDS 2019 - 2nd Int. Conf. Comput. Intell. Data Sci. Proc.*, no. October, 2019, doi: 10.1109/ICCIDS.2019.8862033.

[162] M. Amoozegar and B. Minaei-Bidgoli, "Optimizing multi-objective PSO based feature selection method using a feature elitism mechanism," *Expert Syst. Appl.*, vol. 113, pp. 499–514, 2018, doi: 10.1016/j.eswa.2018.07.013.

[163] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)," *IEEE Access*, vol. 9, pp. 26766–26791, 2021, doi: 10.1109/ACCESS.2021.3056407.

[164] M. Iqbal, M. M. Abid, M. N. Khalid, and A. Manzoor, "Review of feature selection methods for text classification," *Int. J. Adv. Comput. Res.*, vol. 10, no. 49, pp. 138–152, 2020, doi: 10.19101/ijacr.2020.1048037.

[165] G. Ansari, T. Ahmad, and M. N. Doja, “Hybrid Filter–Wrapper Feature Selection Method for Sentiment Classification,” *Arab. J. Sci. Eng.*, vol. 44, no. 11, pp. 9191–9208, 2019, doi: 10.1007/s13369-019-04064-6.

[166] C. Fu *et al.*, “Forwarding behavior prediction based on microblog user features,” *IEEE Access*, vol. 8, pp. 95170–95187, 2020, doi: 10.1109/ACCESS.2020.2995411.

[167] O. Loyola-Gonzalez, R. Monroy, J. Rodriguez, A. Lopez-Cuevas, and J. I. Mata-Sanchez, “Contrast Pattern-Based Classification for Bot Detection on Twitter,” *IEEE Access*, vol. 7, pp. 45800–45817, 2019, doi: 10.1109/ACCESS.2019.2904220.

[168] F. Thabtah, F. Kamalov, S. Hammoud, and S. R. Shahamiri, “Least Loss: A simplified filter method for feature selection,” *Inf. Sci. (Ny).*, vol. 534, pp. 1–15, 2020, doi: 10.1016/j.ins.2020.05.017.

[169] M. Monirul Kabir, M. Monirul Islam, and K. Murase, “A new wrapper feature selection approach using neural network,” *Neurocomputing*, vol. 73, no. 16–18, pp. 3273–3283, 2010, doi: 10.1016/j.neucom.2010.04.003.

[170] Help Center, “Glossary,” *Twitter, Inc.*, 2023. https://help.twitter.com/en/resources/glossary (accessed Apr. 10, 2023).

[171] A. Derhab, R. Alawwad, K. Dehwah, N. Tariq, F. A. Khan, and J. Al-Muhtadi, “Tweet-Based Bot Detection Using Big Data Analytics,” *IEEE Access*, vol. 9, pp. 65988–66005, 2021, doi: 10.1109/ACCESS.2021.3074953.

[172] L. Mandloi and R. Patel, “Twitter sentiments analysis using machine learninig methods,” *2020 Int. Conf. Emerg. Technol. INCET 2020*, pp. 1–5,

2020, doi: 10.1109/INCET49848.2020.9154183.

[173] H. Ping and S. Qin, “A social bots detection model based on deep learning algorithm,” *Int. Conf. Commun. Technol. Proceedings, ICCT*, vol. 2019-Octob, pp. 1435–1439, 2019, doi: 10.1109/ICCT.2018.8600029.

[174] D. Martin-Gutierrez, G. Hernandez-Penaloza, A. B. Hernandez, A. Lozano-Diez, and F. Alvarez, “A Deep Learning Approach for Robust Detection of Bots in Twitter Using Transformers,” *IEEE Access*, vol. 9, pp. 54591–54601, 2021, doi: 10.1109/ACCESS.2021.3068659.

[175] A. Kumar and A. Jaiswal, “Swarm intelligence based optimal feature selection for enhanced predictive sentiment accuracy on twitter,” *Multimed. Tools Appl.*, vol. 78, no. 20, pp. 29529–29553, 2019, doi: 10.1007/s11042-019-7278-0.

[176] I. Jost and J. F. Valiati, “Deep learning applied on refined opinion review datasets,” *Intel. Artif.*, vol. 21, no. 62, pp. 91–102, 2018, doi: 10.4114/INTARTIF.VOL21ISS62PP91-102.

[177] A. Pandya, M. Oussalah, P. Monachesi, and P. Kostakos, “On the use of distributed semantics of tweet metadata for user age prediction,” *Futur. Gener. Comput. Syst.*, vol. 102, pp. 437–452, 2020, doi: 10.1016/j.future.2019.08.018.

[178] H. Khalil, M. U. S. Khan, and M. Ali, “Feature Selection for Unsupervised Bot Detection,” *2020 3rd Int. Conf. Comput. Math. Eng. Technol. Idea to Innov. Build. Knowl. Econ. iCoMET 2020*, pp. 1–7, 2020, doi: 10.1109/iCoMET48670.2020.9074131.

[179] G. A. De Souza and M. Da Costa-Abreu, “Automatic offensive language

detection from Twitter data using machine learning and feature selection of metadata," *Proc. Int. Jt. Conf. Neural Networks*, 2020, doi: 10.1109/IJCNN48605.2020.9207652.

[180] R. R. Rout, G. Lingam, and D. V. L. N. Somayajulu, "Detection of Malicious Social Bots Using Learning Automata with URL Features in Twitter Network," *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 4, pp. 1004–1018, 2020, doi: 10.1109/TCSS.2020.2992223.

[181] F. Wei and U. T. Nguyen, "Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings," *Proc. - 1st IEEE Int. Conf. Trust. Priv. Secur. Intell. Syst. Appl. TPS-ISA 2019*, pp. 101–109, 2019, doi: 10.1109/TPS-ISA48467.2019.00021.

[182] J. Hani, M. Nashaat, M. Ahmed, Z. Emad, E. Amer, and A. Mohammed, "Social media cyberbullying detection using machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, pp. 703–707, 2019, doi: 10.14569/ijacsa.2019.0100587.

[183] L. Ketsbaia, B. Issac, and X. Chen, "Detection of hate tweets using machine learning and deep learning," *Proc. - 2020 IEEE 19th Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2020*, pp. 751–758, 2020, doi: 10.1109/TrustCom50675.2020.00103.

[184] K. Sato, J. Wang, and Z. Cheng, "Credibility Evaluation of Twitter-Based Event Detection by a Mixing Analysis of Heterogeneous Data," *IEEE Access*, vol. 7, pp. 1095–1106, 2019, doi: 10.1109/ACCESS.2018.2886312.

[185] L. Ilias and I. Roussaki, "Detecting malicious activity in Twitter using deep learning techniques," *Appl. Soft Comput.*, vol. 107, p. 107360, 2021, doi: 10.1016/j.asoc.2021.107360.

[186] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Detecting automation of Twitter accounts: Are you a human, bot, or cyborg?,” *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 6, pp. 811–824, 2012, doi: 10.1109/TDSC.2012.75.

[187] “sklearn.preprocessing.StandardScaler — scikit-learn 1.2.2 documentation,” *scikit-learn.org*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (accessed Apr. 18, 2023).

[188] K. Rafay, “Age in days,” *Omni is a Polish Company*, 2022. https://www.omnicalculator.com/everyday-life/age-in-days

[189] “What is a verified Twitter account and how to get one - The Verge.” https://www.theverge.com/23199155/verified-twitter-account-how-to (accessed Nov. 15, 2022).

[190] “What does Twitter verification really mean? And what may happen to it? - MarketWatch.” https://www.marketwatch.com/story/what-does-twitter-verification-really-mean-11667495004 (accessed Jan. 18, 2023).

[191] N. SCHAFFER, “Twitter Following vs Followers: What is the Ideal Ratio?,” *NEAL SCHAFFER*, 2023. https://nealschaffer.com/twitter-followers-following-quality-or-quantity/ (accessed Nov. 16, 2023).

[192] D. Gnanambal, D. Thangaraj, Meenatchi V T, and D. Gayathri, “Classification Algorithms with Attribute Selection: an evaluation study using WEKA,” *Int. J. Adv. Netw. Appl.*, vol. 09, no. 06, pp. 3640–3644, 2018.

[193] K. Lee, B. D. Eoff, and J. Caverlee, “Seven Months with the Devils: An in-

depth characterisation of Bots and Humans on Twitter," *Fifth Int. AAAI Conf. Weblogs Soc. Media*, pp. 185–192, 2011, [Online]. Available: http://arxiv.org/abs/1704.01508

[194] AlindGupta, "Elbow Method for optimal value of k in KMeans," *geeksforgeeks*, 2023. https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/ (accessed May 02, 2023).

[195] A. Ali *et al.*, "Practically Implementation of Information Loss: Sensitivity, Risk by Different Feature Selection Techniques," *IEEE Access*, vol. 10, pp. 27643–27654, 2022, doi: 10.1109/ACCESS.2022.3152963.

[196] Jan Novotny, "Twitter bot Detection & Categorization," 2019.

[197] S. Cresci, F. Lillo, D. Regoli, S. Tardelli, and M. Tesconi, "$FAKE: Evidence of spam and bot activity in stock microblogs on twitter," *12th Int. AAAI Conf. Web Soc. Media, ICWSM 2018*, no. Icwsm, pp. 580–583, 2018.

[198] N. Rahmah and I. S. Sitanggang, "Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 31, no. 1, 2016, doi: 10.1088/1755-1315/31/1/012012.

[199] "KNN Algorithm | Latest Guide to K-Nearest Neighbors." https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/ (accessed Mar. 23, 2023).

[200] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Inf. Sci. (Ny).*, vol. 260, pp. 64–73, 2014, doi: 10.1016/j.ins.2013.11.016.

[201] H. Tajalizadeh and R. Boostani, "A Novel Stream Clustering Framework for

Spam Detection in Twitter," *IEEE Trans. Comput. Soc. Syst.*, vol. 6, no. 3, pp. 525–534, 2019, doi: 10.1109/TCSS.2019.2910818.

[202] M. Mazza, M. Avvenuti, S. Cresci, and M. Tesconi, "Investigating the difference between trolls, social bots, and humans on Twitter," *Comput. Commun.*, vol. 196, no. December 2021, pp. 23–36, 2022, doi: 10.1016/j.comcom.2022.09.022.

[203] H. Elgazzar, K. Spurlock, and T. Bogart, "Evolutionary clustering and community detection algorithms for social media health surveillance," *Mach. Learn. with Appl.*, vol. 6, no. March, p. 100084, 2021, doi: 10.1016/j.mlwa.2021.100084.

[204] M. Z. Ansari, T. Ahmad, and A. Fatima, "Feature Selection on Noisy Twitter Short Text Messages for Language Identification," *Int. J. Recent Technol. Eng.*, vol. 8, no. 4, pp. 10505–10510, 2019, doi: 10.35940/ijrte.d4360.118419.

[205] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," *Processing*, vol., pp. 1–6, 2009.

[206] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The impact of features extraction on the sentiment analysis," *Procedia Comput. Sci.*, vol. 152, pp. 341–348, 2019, doi: 10.1016/j.procs.2019.05.008.

[207] H. Utama, "Sentiment analysis in airline tweets using mutual information for feature selection," *2019 4th Int. Conf. Inf. Technol. Inf. Syst. Electr. Eng. ICITISEE 2019*, pp. 295–300, 2019, doi: 10.1109/ICITISEE48480.2019.9003903.

[208] F. Akba, I. T. Medeni, M. S. Guzel, and I. Askerzade, "Assessment of

iterative semi-supervised feature selection learning for sentiment analyses: Digital currency markets," *Proc. - 14th IEEE Int. Conf. Semant. Comput. ICSC 2020*, pp. 459–463, 2020, doi: 10.1109/ICSC.2020.00088.

[209] M. Wischnewski, A. Bruns, and T. Keller, "Shareworthiness and Motivated Reasoning in Hyper-Partisan News Sharing Behavior on Twitter," *Digit. Journal.*, vol. 9, no. 5, pp. 549–570, 2021, doi: 10.1080/21670811.2021.1903960.

[210] A. Aleroud, N. Abu-Alsheeh, and E. Al-Shawakfa, "A graph proximity feature augmentation approach for identifying accounts of terrorists on twitter," *Comput. Secur.*, vol. 99, p. 102056, 2020, doi: 10.1016/j.cose.2020.102056.

[211] Y. Madani, M. Erritali, J. Bengourram, and F. Sailhan, "A multilingual fuzzy approach for classifying Twitter data using fuzzy logic and semantic similarity," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 8655–8673, 2020, doi: 10.1007/s00521-019-04357-9.

[212] Z. Chen and D. Subramanian, "An Unsupervised Approach to Detect Spam Campaigns that Use Botnets on Twitter," pp. 1–7, 2018, [Online]. Available: http://arxiv.org/abs/1804.05232

[213] S. Barbon, G. F. C. Campos, G. M. Tavares, R. A. Igawa, M. L. Proença, and R. C. Guido, "Detection of human, legitimate bot, and malicious bot in online social networks based on wavelets," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 14, no. 1s, 2018, doi: 10.1145/3183506.

[214] P. A. Chew, *Searching for unknown unknowns: Unsupervised bot detection to defeat an adaptive adversary*, vol. 10899 LNCS. Springer International Publishing, 2018. doi: 10.1007/978-3-319-93372-6_39.

[215] H. Kamioka, “Preferred reporting items for systematic review and meta-analysis protocols (prisma-p) 2015 statement,” *Japanese Pharmacol. Ther.*, vol. 47, no. 8, pp. 1177–1185, 2019.

[216] M. Bibi *et al.*, “Class association and attribute relevancy based imputation algorithm to reduce twitter data for optimal sentiment analysis,” *IEEE Access*, vol. 7, pp. 136535–136544, 2019, doi: 10.1109/ACCESS.2019.2942112.

## Appendix A: Summary of Twitter Datasets Used in Prior Literature

This appendix provides a condensed overview of the Twitter datasets employed in previous studies, serving as a valuable resource for researchers seeking appropriate datasets for future studies. By referring to this summary, researchers can quickly find pertinent information about previous Twitter datasets, enabling them to make informed decisions and select datasets that align with the objectives and requirements of their research projects.

**Table 1: Social Media Bot Detection Datasets - A Comparative Analysis**

| Ref. | Dataset Name | Description | Purpose | |
|---|---|---|---|---|
| [166] | FU, 2020 | The dataset contains information about 10,000 users, with 5,000 users who have forwarded posts and 5,000 users who have not. | Predict the forwarding behavior of microblog users | Labe led |
| [204] | Ansari, 2020 | The dataset contains approximately 6,000 short text messages. | language identification on noisy Twitter short text messages | Labe led |
| [150] | Collected from Kaggle. | The dataset contains 10,000 Twitter profiles, including both genuine and fake profiles. | Detection of fake profiles on Twitter. | Labe led |
| [159] | ABDUR RASOOL,2020 | The dataset contains 4,000 text samples, from various sources, such as social media, product reviews, and news articles which are labeled with positive, negative, or neutral sentiments. | Sentiment analysis | Labe led |
| [205] | Alec Go, 2009 | The dataset contains 1.6 million tweets, which are labeled with positive or negative sentiments based on the presence of emoticons. | Sentiment Classification on Twitter | Labe led |
| [154] | Cresci et al., 2019 | The dataset contains 7,000 Twitter profiles, which are labeled as either real or fake based on manual inspection. | Detecting fake accounts on Twitter | Labe led |
| [172] | Mandloi, 2020 | The dataset contains 10,000 tweets, which are labeled with positive, negative, or neutral sentiments based on the presence of emoticons or keywords. | Sentiment Classification on Twitter | Labe led |

| | | | | |
|---|---|---|---|---|
| [160] | Hajar Rehioui, 2019 | The dataset contains 6,000 tweets, which are labeled with positive, negative, or neutral sentiments based on the presence of emoticons or keywords. | Sentiment Classification on Twitter | Labe led |
| [184] | SATO, 2018 | The dataset contains tweets related to various events, such as the 2016 US Presidential Election, the 2017 Manchester Arena bombing, and the 2018 FIFA World Cup. The credibility scores were assigned by human annotators. | Evaluate the credibility of event detection on Twitter. | Labe led |
| [152] | Mabrook S. Al-Rakhami,2020 | The dataset contains tweets related to COVID-19 that were collected from Twitter between March 2020 and May 2020. labeled as either "misinformation" or "non-misinformation" based on their content. | Detect COVID-19 misinformation on Twitter. | Labe led |
| [206] | Dataset Tweet SS | dataset of movie reviews from the IMDb website | Sentiment analysis | |
| [161] | Stanford sentiment | A total of 140 datasets 1.6 m tweets with two labels, namely, positive and negative. | Sentiment analysis | Labe led |
| [39] | Elakkiya E,2020 | Twitter's API t with over 600 million tweets, including more than 6.5 million spam tweets | Twitter Spam Detection | Labl ed |
| [179] | Gabriel Araujo De Souza, 2019 | Contains 24783 tweets. From these, 1,430 are classified as hate speech, 19,190 as offensive language, and 4,163 as normal language | Offensive language detection from Twitter | Labe led |
| [207] | Public from Kaggle | Tweets.csv(airline-sentiment, 2015, 1.13MB,14641 x 15) | Sentiment analysis on Twitter. | Labe led |
| [180] | Rashmi Ranjan Rout, 2020 | The dataset was collected using Twitter's API and contains 15,000 tweets from 1,000 users, with an equal number of genuine and malicious users. | Detection of Malicious Social Bots | Labe led |
| [208] | Firat AKBA, 2020 | The dataset contains tweets related to different digital currencies, such as Bitcoin and Ethereum, and their sentiments (positive, negative, or neutral). | Sentiment analysis of digital currency markets on Twitter | Labe led |
| [177] | Pandya, 2020 | Three existing datasets Dutch (age 0-40, 2150 users), English1 (age 13-40, 1074 users), English2 (13-25, 1794 users) | Age prediction | |
| [158], [173], [181] | Cresci-2017 | Consisting of 3,474 human accounts 8.4 million and 1,455 bots 3 million tweets. | Detecting malicious Bots on Twitter | Labe led |

| | | | | |
|---|---|---|---|---|
| , [185] | | | | |
| [151] | Murugan, 2018 | The dataset contains 17 million users' tweets with 159 features included | Detecting Twitter spammers | Labe led |
| [46] | Collected Twitter data using Brandwatch . | The dataset contains over 56,000 tweets collected from Twitter using a combination of search terms related to AI and advertising. | Understanding Twitter conversations about artificial intelligence in advertising | |
| [209] | Magdalena Wischnews ki, 2021 | Collected tweets using the Global Database of Events, Language, and Tone, and identified 169 Infowars articles during the period of 23 to 29 September 2019. | Hyperpartisan news-sharing behavior | |
| [210] | Publicly datasets on the Kaggle | The dataset consists of over 1 million tweets collected from Twitter using a combination of search terms related to terrorism and extremist groups. | Identifying accounts of terrorists on Twitter | Labl ed |
| [211] | Youness Madani, 2019 | Use a Twitter API called Twitter4j (between June 2015 and June 2017), The contents of the dataset are not described in detail in the paper. | Classifying Twitter data according to their topics | |
| [167] | Octavio Loyola-Gonzalez, 2019 | 51,457 tweets of which 31,654 belong to humans and the remaining (19,804) belong to bots. | Bot detection on Twitter | Labe led |
| [156] | Pratama, 2019 | Tweets are gathered from the presidential candidates, from February 2019. | Candidate's Supporters | |
| [183] | Dataset of the University of Maryland | The first dataset consists of over 30,000 tweets, whereas the second dataset by Davidson et al., contains roughly 25,000 tweets. | Detecting hate speech | |
| [174] | Twitter API | The dataset is composed of 37438 Twitter accounts, where 25013 were annotated as human accounts and the remaining 12425 are bots. | Bot detection on Twitter | Labe led |
| [212] | Chen2018 | The authors do not describe any specific dataset used in the study. Instead, the authors describe the process they used to collect tweets from Twitter using the Twitter API. | Detect Spam Campaigns on Twitter | Labe led |
| [44] | DeBot | The dataset contains over 16 million tweets collected over several months using Twitter Streaming API. | Bot detection on Twitter | Labe led |

| | | | | |
|---|---|---|---|---|
| [147] | Beskow2019 | The dataset was collected using the Twitter API and Twitter accounts that were manually labeled as bots or humans. | Bot detection on Twitter | Labeled |
| [213] | Campos2018 | The dataset consists of 7,500 Twitter users manually labeled as either humans, legitimate bots, or malicious bots, and includes a total of 100,000 tweets. | Detection of humans, legitimate bots, and malicious bot on Twitter | Labeled |
| [214] | Chew2018 | The dataset used in the study consists of 10 million tweets posted by over 1 million Twitter users. | Bot detection on Twitter | |

# Appendix B: Common Features Used for Social Bot Detection on Twitter

**Table 1: The important features used in the previous studies for bot detection**

| Ref. | Feature | Description | Taxonomy | Type |
|---|---|---|---|---|
| [11], [150], [216], [151], [152], [157], [166], [177], [180], [207], [215] | FolloweeFollower | Mean of the no. of followers of a user's followers | Metadata | Account information |
| | FolloweeFollowerMedian | The median of the no. of followers of a user's followers | Metadata | Account information |
| | FolloweeFollowerStdDev | Deviation of the no. of followers of a user's followers | Metadata | Account information |
| | FolloweeFollowerEntropy | The entropy of the no. of followers of a user's followers | Metadata | Account information |
| | MentionCountRatio | No. of mentions or total no. of tweets | Metadata | Account information |
| | MentionUniqueRatio | No. of unique mentions/total no. of mentions | Metadata | Account information |
| | ffratio | Friends-to-followers ratio | Metadata | Account information |
| | listed | Number of listed tweets in the account | Metadata | Account information |
| | Volume of tweeting | One spam indicator is unusually high-volume tweeting, which is often bot-generated. This could be measured by a raw count of tweets or the percentage of tweets posted to a hashtag by a single user. | Metadata | Account information |
| | Friends_Count | The number of users this account is following | Metadata | Account information |
| | AccountBackground | Whether the user profile has a background image | Metadata | Account usage |
| | AccountSourceTweets | Whether the user is the source tweet's author | body of tweet | Account usage |

| | | | | |
|---|---|---|---|---|
| | URL in profile | True if a URL is specified in the account's profile | Metadata | Account usage |
| | has biography | True if the biography is specified in the account's profile | Metadata | Account usage |
| | Retweets | The ratio between retweet count and tweet count | Metadata | Account usage |
| | Replies | The ratio between the reply count | Metadata | Account usage |
| | Favorite | The ratio between the favorite tweet and tweet count | Metadata | Account usage |
| | Hashtag | The ratio between hashtag count and tweet count | Metadata | Account usage |
| | URL | The ratio between URL count and tweet count | Metadata | Account usage |
| | Favorites | Number of tweets favorited in this account | Metadata | Account usage |
| | Language_Code | The BCP 47 code for the user's self-declared user interface language.Justification: Fake accounts tend to have different language codes on their interface. | Metadata | Account usage |
| | Char_account | No. of characters in the user's name including white space | Metadata | Account usage |
| | Coordinates | It represents the geographic location of the tweet. | Metadata | Location |
| | AccountAge | Total duration from since account created till now | Metadata | Temporal |
| | AverageTweetCount | No. of tweets/account age | Metadata | Temporal |
| | ProbWeekend | Probability of a user tweeting on the weekend | Metadata | Temporal |
| | ProbMorning | Probability of a user tweeting in the morning | Metadata | Temporal |
| | ProbAfternoon | Probability of a user tweeting in the afternoon | Metadata | Temporal |
| | ProbEvening | Probability of a user tweeting in the evening | Metadata | Temporal |
| | ProbNight | Probability of a user tweeting at night | Metadata | Temporal |
| | Hour-x | Probability of a user tweeting at hour x | Metadata | Temporal |
| | Weekday-x | Probability of a user tweeting on day x | Metadata | Temporal |
| | intertime | Average seconds between postings | Metadata | Temporal |

| | | | | |
|---|---|---|---|---|
| | id_created days | No of days id created | Metadata | Temporal |
| | tweet_year | The year when the tweet was created | Metadata | Temporal |
| | tweet_month | The month when the tweet was created | Metadata | Temporal |
| | tweet_day | The day when the tweet was created | Metadata | Temporal |
| | tweet_hour | The hour when the tweet was created | Metadata | Temporal |
| | user_created_year | The year when the Twitter account was created | Metadata | Temporal |
| | user_created_month | The month when the Twitter account was created | Metadata | Temporal |
| | user_created_day | The day when the Twitter account was created | Metadata | Temporal |
| | user_created_hour | The hour when the Twitter account was created | Metadata | Temporal |
| | The count of total words in a tweet; | | Body of tweet | Tweet-level features |
| | The count of exclamations; | | Body of tweet | Tweet-level features |
| | Entity extraction | URLs or hashtags, particularly photos, etc | Body of tweet | Tweet-level features |
| | Twitter hashtag features | No. of tweets that have at least one hashtag, Avg no. of hashtags per tweet | Body of tweet | Tweet-level features |
| | Agreement | Whether the text has an agreement text. | Body of tweet | Tweet-level features |
| | HashtagAve | Hashtag count / Tweet count | Body of tweet | Tweet-level features |
| | LinkAve | Link count / Tweet count | Body of tweet | Tweet-level features |

## الخلاصة

يواجه Twitter تحدي هجمات الروبوتات، والتي يمكن أن تؤثر على المجتمع. يعد الحصول على البيانات المصنفة لتقنيات التعلم الآلي الخاضعة للإشراف التي تكتشف الروبوتات أمرًا مكلفًا ويستغرق وقتًا طويلاً. لذلك، تقترح هذه الرسالة تقنية تجميع غير خاضعة للإشراف لاكتشاف الروبوتات بدقة على توتر باستخدام بيانات غير مسماة. الهدف هو توفير نظام موثوق وفعال لاكتشاف الروبوتات، وتخفيف انتشار المعلومات الخاطئة، وتحسين المحادثات عبر الإنترنت. تعالج التقنية المقترحة التحدي المتمثل في توافر البيانات المصنفة وتسعى إلى تحقيق توازن بين الدقة والتعقيد الحسابي أثناء التعامل مع القيم المتطرفة وتحسين استخدام الذاكرة.

تقترح الأطروحة نظامًا آليًا يمكنه اكتشاف روبوتات Twitter والتي تتضمن عمومًا جزأين رئيسيين: توليد ميزات جديدة ونهج تنبؤات الروبوتات. على وجه الخصوص، يتكون النظام المقترح من خمس مراحل: المعالجة المسبقة، وتحليل الميزات، وتصميم التجربة، وتطوير التقنيات، وتقييم النظام. يتم تطوير الميزات المقترحة باستخدام مجموعة من الأساليب الإحصائية واليدوية، ويتم اختيار الميزات ذات التصنيف الأعلى فقط باستخدام تقنية تقييم سمة الارتباط. علاوة على ذلك، تم تعديل ثلاث خوارزميات غير خاضعة للإشراف للتعلم الآلي وهي KNN وDBSCAN وK-mean Stream للعمل تلقائيًا لاكتشاف الروبوت عن طريق تحديد معلمات الإدخال لهذه الخوارزميات بناءً على خصائص البيانات الفريدة.

لتقييم أداء الميزات المقترحة، تم استخدام ثلاثة أنواع من خوارزميات التجميع: التقسيم ( k-mean، k-medoid) ، التسلسل الهرمي (التكتلي) ، والكثافة (DBSCAN) للكشف عن الروبوتات. تظهر النتائج أن الميزات المقترحة أكثر فعالية من الميزات الأصلية لاكتشاف الروبوتات. يعلن نجاح هذا الاختبار للميزات المقترحة عن بدء الجزء الثاني من نظامنا وهو تحديد الروبوتات باستخدام خوارزميات التجميع المعدلة.

تم تأكيد فعالية التقنية المقترحة في اكتشاف روبوتات توتر باستخدام خوارزميات التجميع، وخاصة خوارزميات التجميع القائمة على التدفق، في هذه الرسالة. يتم استخدام مقاييس تقييم متعددة حيث يتم ملاحظة القيم العالية لمقياس التجانس (0.988)، مقياس الاكتمال (0.989)، مقياس V (0.989) ، مقياس مؤشر Rand المعدل (0.996) ، مقياس المعلومات المتبادلة المعدلة (0.989) ، مقياس معامل Silhouette (0.786) ومقياس Fowlkes Mallows (0.998).

تؤكد هذه الرسالة على أن استخدام تقنيات التجميع يعد حلاً فعالاً من حيث التكلفة لتصنيف بيانات Twitter غير المسماة. يمكن أن تقدم مثل هذه الأساليب نهجًا عمليًا لتحديد روبوتات Twitter. يسلط البحث الضوء على أهمية استخراج السمات وتقليل الأبعاد في تحسين أداء خوارزميات التجميع. يوفر دمج خوارزميات متعددة غير خاضعة للإشراف نهجًا أكثر شمولاً لاكتشاف روبوتات Twitter. يساهم هذا البحث في مجال اكتشاف الروبوتات وتحليلات الوسائط الاجتماعية، مما يوفر إمكانيات جديدة لتحليل بيانات Twitter.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنلوجيا المعلومات
قسم البرمجيات

# اكتشاف روبوت التويتر باستخدام خوارزميات التجميع المحسنة

**أطروحة مقدمة الى**

**مجلس كلية تكنولوجيا المعلومات ـ جامعة بابل كجزء من متطلبات نيل درجة الدكتوراه فلسفة في تكنولوجيا المعلومات / البرمجيات**

**من قبل**

رائد غازي حميد كريم

**باشراف**

أ.د صفاء عبيس مهدي مظلوم

**1445 هـ** **2023 مـ**