Available online at www.garmian.edu.krd

Passer Journal of Basic and Applied Sciences

Passer

Passer 8 (Issue 2) (2026) 817-831

Original Article

Real-Time Bot Detection on XCorp: Enhanced Feature Engineering with Apache Spark

Hakar Mohammed Rasul [1], Alaa Khalil Jumaa [2], Aysar Abdulrahman * [3]

[1] *Software Engineering Department, Faculty of Engineering and Computer Science, Qaiwan International University, Sulaymaniyah 46001, Kurdistan Reign, Iraq.*

[2] *Database Technology Department, Technical College of Informatics, Sulaimani Polytechnic University, Sulaymaniyah 46001, Kurdistan Reign, Iraq.*

[3] *Computer Networks Department, Technical College of Informatics, Sulaimani Polytechnic University, Sulaymaniyah 46001, Kurdistan Reign, Iraq.*

Received 24 January 2026; revised 17 May 2026;
accepted 09 June 2026; available online 20 July 2026

DOI: 10.24271/PSR.2026.572198.2565

ABSTRACT

The rapid growth of misinformation, spam, and automated bot activity on social media platforms such as X (formerly Twitter, operated by XCorp) has created a pressing need for accurate and real-time detection systems. Bots are one of the most important challenges at XCorp. A bot on XCorp is software that sends tweets to users automatically. XCorp bots may be used for several advantageous purposes or may be created for malicious purposes such as spreading fake news campaigns, spamming, and violating the privacy of others. This study presents a scalable framework for real-time tweets analysis using Apache Spark, focusing on the integration of user-based features and context-based features to enhance bot detection performance. Three distinct datasets were utilized to build and train the proposed model, enabling robust feature learning and generalization across different data sources. A large stream of real-time XCorp data was then processed using Spark Streaming, where newly engineered features captured user and content features within tweets. The model was constructed using five PySpark machine learning techniques: Gradient Boosted Trees Classification, Linear Support Vector Machine Classification, Decision Tree Classification, Logistic Regression, and Random Forest Classification. Experimental results show that incorporating the proposed features significantly improves accuracy, precision, and recall compared with baseline models and existing studies, while maintaining a relatively stable computation time. Among the evaluated models, the Random Forest classifier achieved the best performance, reaching 99.1% precision, 99.7% recall, 99.40% F1-Score and 99.6% accuracy.

Keywords: XCorp bot detection, real-time tweet analysis, machine learning, Apache Spark, user-based features, content-based features, sentiment analysis.

1 Introduction

Social media analytics refers to the process of collecting and analyzing data from social platforms to support decision-making and evaluate the effectiveness of online activities. X Corp. (formerly known as Twitter) is one of the most widely used social networking platforms, enabling users to post short messages called tweets, follow other users, and interact in real time. In this paper, we refer to the platform as XCorp throughout. Users can follow accounts of interest and engage with content through actions such as tweeting, replying, and retweeting, allowing information to spread rapidly across the network. During live events such as sports matches or award ceremonies, user activity often increases significantly, resulting in spikes in tweet volume. Tweets are limited in length, and hashtags (#) are commonly used to categorize content and facilitate topic tracking. Most user accounts are public, allowing anyone to view their posts, while private accounts restrict access to approved followers [1]. One of the most serious issues on XCorp is the presence of bots. Users receive many tweets, some of which are generated by bots. A bot on XCorp is software that automatically sends tweets, often performing tasks such as spamming. Bot detection is a crucial solution to address this issue, as it helps identify fake accounts and protect legitimate users from misinformation and malicious activity [2]. Recent studies emphasize the importance of real-time social media data analysis for understanding user behavior and detecting malicious activities. A survey of real-time Twitter data analysis techniques highlights the use of streaming APIs, data preprocessing, and machine learning approaches for extracting insights from large-scale tweet streams. The study also emphasizes the challenges associated with handling high-velocity data and discusses the role of real-time analytics in applications

* Corresponding author
E-mail address aysser.abdulrahman@spu.edu.iq (Instructor).
Peer-reviewed under the responsibility of the University of Garmian.

such as trend monitoring and the detection of anomalous behaviors, including spam and bot activity [3]. Given the large volume of real-time XCorp data, Apache Spark is used for processing. Spark is a fast engine for large-scale data processing that performs in-memory computation and supports efficient iterative tasks such as machine learning. It has been widely reported to outperform traditional MapReduce frameworks, such as those used in Apache Hadoop, particularly for iterative workloads [4]. Spark also enables parallel processing across multiple nodes, providing a unified engine capable of handling a wide range of data challenges. It supports programming languages such as Scala, Java, Python, and R. Machine learning (ML) is a key component of Spark, encompassing techniques such as classification, regression, clustering, collaborative filtering, and model evaluation [5]. Recent research highlights the growing role of artificial intelligence in security-related applications, including the detection of malicious activities using machine learning and deep learning techniques. AI-driven approaches have been widely adopted in areas such as intrusion detection and threat analysis, emphasizing the importance of scalable and real-time data processing frameworks. These trends align with the need for efficient feature engineering and distributed processing in bot detection systems, where platforms such as Apache Spark enable the analysis of large-scale data streams for identifying suspicious behaviors [6]. This paper focuses on developing an efficient model for detecting and classifying fake bot accounts on XCorp by leveraging multiple factors, including user-based features, tweet content features, and sentiment score features, with a particular emphasis on real-time bot detection. The primary contribution of this work is a model designed to identify bots involved in promoting specific hashtags using real-time data. The novelty of the approach lies in combining user-based features with content-based features derived from tweet sentiment, including polarity and subjectivity, to improve the detection of automated accounts. The proposed system consists of several stages, including data collection, preprocessing, sentiment analysis, feature extraction, and classification. To support real-time processing, scalable data storage and analysis tools such as Apache Spark are employed, along with XCorp API functions for collecting streaming data. The XCorp Streaming API enables access to sampled and filtered tweets in real time, including responses and mentions from public accounts, allowing continuous monitoring of user activity. Access to the API requires a developer account, which provides authentication credentials such as API keys and access tokens for secure data collection [7].

The paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the key concepts and tools used in this study, including bots on XCorp, sentiment analysis with TextBlob, and Apache Spark using PySpark. Section 4 presents the baseline dataset. Section 5 describes the proposed system and methodologies. Section 6 presents the implementation of the proposed model along with results, analysis, and discussion. Finally, Section 7 concludes the paper, highlighting the contributions and suggesting potential directions for future work

2 Related Works

Research on social media bot detection has employed a variety of techniques, features, and datasets to identify automated accounts. Prior studies have focused on user behavior, content characteristics, and graph-based attributes to distinguish between human and bot accounts. The following paragraphs summarize key contributions relevant to our work and highlight the features, methodologies, and performance reported in each study.

To detect spam bots, Wang (2010) employed three graph-based and three tweet-based features. The graph-based features such as the number of friends, followers, and follower ratio are extracted from the user's social network, while the tweet-based features such as the number of duplicate tweets, presence of HTTP links, and replies/mentions derived from the user's most recent 20 tweets. The dataset used to evaluate this approach included 25,847 users, approximately 500K tweets, and around 49 million followers/friends, all obtained from publicly available Twitter data. Several classification algorithms were tested, including Decision Tree (DT), Neural Network (NN), Support Vector Machines (SVM), Naive Bayes (NB), and k-Nearest Neighbors (k-NN). Among these, the NB classifier achieved the best results, with 91% accuracy, 91% recall, and 91% F-measure [8].

Zi Chu et al. (2012) classified Twitter users into three groups human, bot, and cyborg based on attributes derived from tweeting behavior, tweet content, and account properties. They hypothesized that bot behavior is less sophisticated than human behavior and used entropy rates to measure process complexity: low rates indicated regular processes, medium rates indicated moderate complexity, and high rates indicated random processes. Tweet content was analyzed to generate text patterns representing recognized spam on Twitter, while account-related features such as the percentage of external URLs, link safety, and account registration date were also included. A RF classifier was applied to evaluate these features and classify users. The approach was tested on a dataset of 500,000 Twitter accounts and achieved an average true positive rate of 96.0% [9].

Cresci et al. (2015) investigated the detection of fake Twitter followers in their work Fame for Sale: Efficient Detection of Fake Twitter Followers. They developed a baseline dataset of verified human and fake follower accounts, combining the HUM (1,950 human accounts, 2,631,730 tweets) and FAK (1,950 fake accounts, 118,327 tweets) datasets to form a total of 3,900 accounts with 2,750,057 tweets, 1,819,991 followers, and 1,788,515 friends. The study also included additional datasets such as TFP, E13, FSF, INT, and TWT, which varied in account numbers, tweets, followers, and friends. Various features and rules proposed by both academia and media were evaluated for identifying anomalous accounts. Their results showed that many media-based rules performed poorly, while academically proposed features achieved strong outcomes. By refining classifiers to reduce overfitting and data collection costs, they proposed a novel Class A classifier that was both lightweight and robust, achieving over 95% accuracy in correctly classifying accounts. This work provides an important foundation for

research on distinguishing genuine users from bots on social media platforms [10].

In their study, Gilani et al. (2017) classified Twitter accounts into two categories: automated bots and real users. They collected data using their own platform, Stweeler, gathering approximately 2.5 to 3 million tweets per day. The dataset was divided into four subsets (10 million, 1 million, 100 thousand, and 1 thousand), each representing different levels of account popularity based on the number of followers. For the labeling process, human annotation was employed, and Cohen's kappa coefficient was used to ensure annotation reliability. In total, 3,536 accounts were used in the testing phase across the four subsets. The authors extracted 15 features and applied an RF classifier after statistical analysis. They conducted five-fold cross-validation using three different experimental setups. The results showed an accuracy of 86.44%, precision of 85.44%, recall of 82.24%, and an F-measure of 83.4%. Among the extracted features, six were identified as the most significant. However, this study has some limitations. First, it relies on human annotation, which may introduce subjectivity. Second, it does not incorporate content-based features, and the use of natural language processing (NLP) techniques for content analysis could potentially improve the system's performance [11].

Cresci et al. (2017) extensively investigated the emergence of a new generation of social spambots on Twitter in their work "The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race." They analyzed the effectiveness of Twitter, human annotators, and state-of-the-art detection techniques in distinguishing between genuine users, traditional spambots, and social spambots. The datasets used in their study were well-structured and included 3,474 genuine (human) accounts, three groups of social spambots containing 991, 3,457, and 3,464 accounts, respectively, as well as 1,000 and 100 traditional spambots. In addition, a dataset of 3,351 fake follower accounts was considered. The authors also tracked account statuses, such as active, deleted, or suspended, to provide deeper insights into bot behavior. Their findings highlighted the limitations of existing detection methods and emphasized the need for more advanced approaches. Notably, these datasets were made publicly available, and a portion of them is utilized in this study for evaluation. Using these datasets, their method achieved approximately 92% accuracy, demonstrating its effectiveness for classification tasks and providing a strong foundation for our work [12].

Lacopo Pozzana et al. (2018) examined four tweet-level metrics to analyze bot behavior during a single activity session: the number of mentions per tweet, tweet length, the percentage of retweets, and the fraction of replies. Their analysis revealed behavioral differences between human users and bots that could be leveraged to improve bot detection algorithms. For instance, human users are consistently exposed to posts from others, increasing opportunities for social interaction. The authors applied five machine learning methods, DT, Extra Trees (ET), RF, Adaptive Boosting (AB), and k-NN to classify tweets as originating from bots or humans. The dataset included over 16 million tweets from more than 2 million unique users. ET and RF achieved the highest cross-validated performance (86%), followed by DT and AB (83%), and k-NN (81%). One limitation of the study is that it did not distinguish whether the detected bots were harmful or benign [13].

Monica and Nagarathna (2020) developed a model to analyze user sentiment in real time for individuals writing about specific topics, using content-based features to detect Twitter spam. Their method applied a custom rule-based algorithm for bot detection and compared it with classifiers including Multi-Layer Perceptron (MLP), DT, and RF. Real-time data were collected via the Twitter API, followed by feature extraction, preprocessing, and sentiment analysis. The rule-based algorithm achieved 97% accuracy, outperforming the other machine learning classifiers. The study had two main limitations: a small user sample and analysis restricted to English-language tweets [14].

In 2022, Anisha et al. developed a system to assess tweet sentiment and classify tweets as spam or ham. After preprocessing, features were extracted and evaluated using multiple classifiers, including DT, Logistic Regression (LR), Multinomial and Bernoulli Naive Bayes, SVM, and RF for spam detection. For sentiment analysis, they also applied deep learning models such as Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), and One-Dimensional Convolutional Neural Network (1D CNN). Two separate datasets were used for spam detection and sentiment analysis. For spam classification, multinomial NB achieved 97.78% accuracy, and LSTM achieved 98.74% validation accuracy. For sentiment analysis, SVM reached 70.56% accuracy, while LSTM achieved 73.81% validation accuracy. These results demonstrate the effectiveness of the extracted features for both spam detection and sentiment estimation [15].

Abdulhammed et al. (2022) present a sentiment analysis framework for social media text classification using machine learning techniques [X]. Their work focuses on extracting sentiment information from Twitter data collected via the Twitter API, where tweets related to topics such as public figures, companies, and events are analyzed after preprocessing. The study applies sentiment scoring methods and multiple feature extraction techniques, including TF-IDF and distributed representation models, to transform textual data into meaningful feature vectors. A Support Vector Machine (SVM) classifier is then used to perform sentiment classification based on the extracted features. Furthermore, the model performance is enhanced using a Shark Smell Optimization (SSO) algorithm to tune the SVM parameters, leading to improved classification accuracy. The experimental results show that the proposed approach achieves an accuracy of **92.12% after optimization**, compared to **88.69% without optimization**, demonstrating the effectiveness of feature engineering and optimization techniques in improving sentiment classification performance on social media data [16].

In 2023, Lei et al. proposed a Twitter bot detection framework that relies solely on textual content using a bidirectional lightweight gated recurrent unit (BiLGRU) combined with multiple linguistic embeddings, including word, character, part-of-speech (POS), and named entity (NE) embeddings. Unlike traditional methods, their model does not require handcrafted features, prior knowledge of user profiles, friendship networks, or historical activity. The experiments were conducted on datasets comprising human accounts (3,474 accounts, 2,839,361 tweets) and two types of social bots: Social-bot-1 (991 accounts, 1,610,034 tweets) and Social-bot-3 (464 accounts, 1,418,557 tweets). Test sets were created as mixed sets of 50% human accounts and 50% bots, with 1,982 accounts for Test Set #1 and 928 accounts for Test Set #2. The study evaluated the impact of combining different embeddings on classification performance. For Test Set #1, the best results were achieved using all four embeddings, with precision 0.956, recall 0.977, accuracy 0.969, and F1-score 0.967. Similarly, for Test Set #2, the complete embedding combination yielded precision 0.938, recall 0.934, accuracy 0.939, and F1-score 0.936. These results demonstrate that leveraging multiple linguistic embeddings within a BiLGRU architecture can achieve high accuracy and generalization in bot detection without relying on external features, making the method efficient for real-world applications [17].

Wei et al. (2024) propose BotGSL, a framework for Twitter bot detection that leverages graph structure learning to model complex user interactions. The method constructs a heterogeneous user graph incorporating both explicit and implicit relations, extracts multiple features to characterize user intent, and fuses relation graphs before embedding them with a Graph Transformer. This integrated representation enables the model to automatically capture structural cues that distinguish human and bot behavior and offers robustness to noise and incomplete data. BotGSL was evaluated on three real-world benchmark Twitter bot detection datasets, which in the graph-based bot detection literature typically include large benchmarks such as TwiBot-20 and TwiBot-22, alongside other public datasets used for comparative evaluation. Across these benchmarks, BotGSL achieved an average detection accuracy of 91.92%, outperforming existing baselines and highlighting the effectiveness of graph structure learning for modeling complex user relationships [18].

In 2025, Anshul et al. proposed RoGBot, a multimodal graph-based bot detection framework that integrates transformer-based semantic embeddings with behavioral and metadata features while leveraging GraphSAGE to capture structural patterns. Unlike traditional graph-based methods, RoGBot does not require explicit social links such as follower-following relationships, making it effective in scenarios where network graphs are incomplete or unavailable. The model extracts deep semantic embeddings from user-generated content, aggregates them using max pooling to form comprehensive user-level representations, and combines these with auxiliary behavioral and metadata features. GraphSAGE then captures both local and global interaction patterns, enabling the detection of subtle, coordinated bot behaviors. RoGBot was evaluated on Cresci-15, Cresci-17, and PAN 2019 datasets, achieving 99.8%, 99.1%, and 96.8% accuracy, respectively. These results highlight the effectiveness of combining semantic, behavioral, and structural information in a unified framework, demonstrating robust performance against increasingly sophisticated social bots [19].

In 2025, Luo and Jin proposed MM-HGT-Bot, a multi-modal social bot detection framework that combines Heterogeneous Graph Transformer (HGT) models with both user content and relational data. The model explicitly decomposes social ties into two edge types: information source selection (following network) and potential influence (follower network). These relational patterns are learned via HGT and fused with context-aware text embeddings from user-generated content, enabling the model to capture subtle anomalies in bot behavior. The framework was evaluated on the Cresci-15 dataset (5,301 users; 3,351 bots) and TwiBot-20 dataset (229,580 users; 6,589 bots), achieving 98.04% accuracy and 98.43% F1-score on Cresci-15 and 84.62 accuracy and 86.96% F1-score on TwiBot-20. These results demonstrate that combining fine-grained relational modeling with content information significantly enhances social bot detection performance [20].

While existing studies have explored different techniques on XCorp, most prior work focuses on offline datasets, limited feature sets, or small-scale experiments that do not operate in real time. Many approaches rely on a single dataset, limiting generalization across user behaviors and tweet content. In contrast, our work addresses these gaps by combining user-based and content-based features, training on three diverse datasets, and implementing the system in a real-time framework, providing a scalable, feature-enriched model capable of live bot detection with improved performance.

3 Preliminaries

3. 1 Bots on XCorp

In contrast to accounts managed by human users, a bot (short for robot), also referred to as a social bot, social media bot, or Sybil account, is a software-controlled account that performs automated actions on social platforms, such as posting, liking, replying, or following, without direct human intervention. These automated agents range in sophistication from simple rule-based scripts to advanced accounts driven by artificial intelligence that mimic human behavior and language patterns. Bots can serve various purposes, including benign activities such as automated news updates or content recommendation, as well as malicious activities. Malicious bots are often designed to spread misinformation, amplify propaganda, manipulate trends, or inflate follower counts, which can distort user perceptions and engagement on social platforms. Fake bot accounts, in particular, are deliberately created to disseminate unwanted or deceptive information, frequently exhibiting repetitive posting patterns, high activity rates, and coordinated behavior across networks. On platforms like XCorp, these bots can significantly compromise the authenticity and reliability of online interactions, making detection and mitigation an important research problem [21].

3. 2 Sentiment Analysis Using TextBlob

Sentiment analysis, often referred to as opinion mining, is an NLP technique used to detect the emotional tone or attitude expressed in textual data. This approach enables the systematic categorization of text into positive, negative, or neutral sentiment, providing valuable insights for understanding user opinions, reactions, and behaviors. In the context of XCorp, sentiment analysis is particularly useful for examining tweets to monitor public opinion, assess customer feedback, and support applications such as social media monitoring, reputation management, and customer experience evaluation [21]. A key aspect of sentiment analysis is the measurement of polarity and subjectivity. Polarity represents the overall emotional tone of a text, phrase, or word, typically expressed as a numerical score ranging from -1 to 1, where -1 indicates negative sentiment, 1 indicates positive sentiment, and 0 reflects neutral sentiment. Polarity can be calculated for an entire text or specific phrases to provide a fine-grained understanding of emotional content. Subjectivity measures the degree to which a text contains personal opinions versus objective facts, with values ranging from 0 (fully objective) to 1 (fully subjective). By analyzing both polarity and subjectivity, sentiment analysis offers a richer understanding of textual emotion and opinion [22]. A sentiment analysis approach has been proposed to classify textual data associated with images by identifying users' emotions and opinions. The model applies machine learning algorithms such as Naive Bayes, Logistic Regression, and Support Vector Machine (SVM), where results show that SVM achieves higher accuracy while Logistic Regression provides faster execution time. This demonstrates the effectiveness of machine learning techniques in analyzing text linked to visual content, which can support enhanced interpretation within real-time food detection systems [23].

TextBlob is a Python NLP library built on the Natural Language Toolkit (NLTK) that provides tools for lexicon-based sentiment analysis. It evaluates the sentiment of text by assigning polarity scores to individual words or phrases and then aggregating them, typically through arithmetic averaging, to determine the overall sentiment of a sentence or text block. TextBlob also incorporates semantic markers, such as emoticons, emojis, punctuation, and exclamation marks, which allow for a more nuanced and accurate analysis of sentiment. The library returns both polarity and subjectivity scores, enabling comprehensive evaluation of emotional content in textual data [24].

By leveraging TextBlob for sentiment analysis, researchers can efficiently process and quantify textual data to reveal underlying opinions, emotions, and trends, making it an essential tool for social media analytics and automated bot detection.

3. 3 Apache Spark and PySpark Framework

Apache Spark is an open-source framework for large-scale data processing in memory (RAM). It leverages distributed memory to process vast amounts of data efficiently, caching datasets across multiple parallel processes. This design makes Spark particularly suitable for iterative algorithms and parallel processing of distributed data. Spark can perform multi-threaded lightweight tasks within Java Virtual Machine (JVM) processes, enabling fast job startup and effective use of multi-core CPUs. Its tasks run in parallel in memory and only spill to local disk when memory limits are reached. Additionally, Spark employs a data pipeline mechanism, which simplifies the management of multiple operations and improves execution efficiency. These capabilities make Spark highly suitable for large-scale machine learning, graph analytics, and other big data applications. Several major companies, including Yahoo, Baidu, and Tencent, rely on Spark for large-scale data processing [4].

PySpark is the Python API for Spark, developed by the Apache Spark project to facilitate Python integration. PySpark allows users to easily create, transform, and manipulate Resilient Distributed Datasets (RDDs) and DataFrames using Python, providing a convenient interface for data engineers and researchers working with large datasets [25]. PySpark combines the power of Spark's distributed in-memory processing with Python's ease of use, enabling efficient computation and analysis of massive datasets.

Some of the key features of PySpark, as summarized by Singh [25], include:

1. In-memory processing: Reduces latency for real-time computations, making PySpark highly efficient for iterative and streaming tasks.
2. Multi-language compatibility: Supports Scala, Java, Python, and R, providing flexibility and widespread adoption in data engineering and analytics workflows.
3. Robust caching and disk persistence: Ensures data reliability and allows repeated computations without re-computation of the entire dataset.
4. High performance for big data: PySpark's architecture and distributed processing make it significantly faster than traditional frameworks when handling large-scale datasets.

Overall, PySpark combines speed, scalability, and flexibility, making it an essential tool for big data processing and analytics, particularly in applications such as real-time bot detection and social media analysis.

Together, these concepts establish the technical foundation for our study. Automated bots represent the target entities to be detected, while sentiment analysis provides insights into the emotional content of user-generated text. TextBlob serves as the primary tool for extracting polarity and subjectivity from textual data, and Apache Spark with PySpark enables efficient processing and analysis of large-scale datasets. Collectively, these components create a robust framework that supports the real-time detection and classification of automated accounts, forming the basis for the methodology described in the following sections.

4 Baseline Datasets

Three datasets from the MIB dataset repository [26] were collected for this study. As explained in the proposed system section, besides analyzing XCorp accounts' features, this study also analyzes the tweets that have been posted by each account in order to calculate the polarity and subjectivity scores for each account. Therefore, only datasets that contain both account features and the tweets posted by each account were selected. For this reason, each of the three datasets used in this study consists of two CSV files. The first file contains the accounts and their features, while the second file contains all tweets posted by those accounts. Table 1 summarizes the datasets, including their names, number of users, number of user features, number of human and bot accounts, number of tweets, number of tweet features, and dataset size (in MB).

The first dataset (**Genuine Accounts)**, which is used by [12], consists of 3,474 XCorp accounts and 8,377,522 tweets. All accounts in this dataset are created and maintained by human users. The data is provided in two separate CSV files. The accounts' CSV file consists of 3,474 rows, each representing a user. Each row contains 31 features providing information about the user. Table 2 lists the exact names of these features along with their descriptions and availability. The features labeled as *deprecated* in the table refer to attributes that are no longer supported by the XCorp platform. The second CSV file contains the tweets posted by the XCorp accounts listed in the first file. The tweets' CSV file consists of 8,377,522 rows, each representing a tweet and containing 17 features describing the tweet.

The second dataset (**Social Spambots #2)**, which is used by [10] and [12], consists of 3,457 XCorp accounts and 428,542 tweets. The structure of this dataset is the same as the first dataset. However, unlike the first dataset, it contains both human and bot accounts, where 191 accounts belong to human users and 3,266 accounts are labeled as bot accounts.

Considering only the first and second datasets would result in an imbalance between human and bot accounts. Therefore, a third dataset (**Social Spambots #3)** from the same MIB repository [26] which is used by [12], was also included. This dataset consists of 464 accounts and 1,418,626 tweets, where 48 accounts belong to human users and 416 accounts are labeled as bot accounts. The dataset is also provided in two CSV files with the same structure and features as the previous datasets.

By combining the three datasets, the final dataset used in this study contains 3,713 human accounts and 3,682 bot accounts, resulting in a nearly balanced distribution between the two classes.

Table 1: Description of the Datasets.

No.	Dataset names	No. of Users	No. of User's Features	No. of Human Accounts	No. of Bot accounts	No. of Tweets	No. of Tweets' Features	Dataset Size in MB
1	**Genuine Accounts (Humans Accounts)**	3474	31	3474	0	8377522	17	966
2	**Social Spambots #2 (Spammers of paid mobile devices)**	3457	31	191	3266	428542	17	141
3	**Social Spambots #3 (spammers of Amazon.com - sold items)**	464	31	48	416	1418626	17	594

Table 2: Features of XCorp Accounts [25].

No.	Column Name	Meaning	Availability	No	Column Exact Name	Meaning	Availability
1	Id	User ID	Available	17	profile_banner_url	URL of the user's profile banner image	Available
2	Name	Display name on the profile.	Available	18	profile_background_image	URL of the user's background image	Deprecated
3	screen_name	Username shown after @.	Available	19	profile_text_color	Defines the webpage text color.	Deprecated
4	statuses_count	Total number of tweets.	Available	20	profile_image_url_https	Secure (HTTPS) URL of the profile picture.	Available
5	followers_count	Number of followers.	Available	21	profile_sidebar_border	Color of the profile sidebar border.	Deprecated
6	friends_count	Number of followings.	Available	22	profile_background_tile	Binary flag indicating if background is tiled.	Deprecated
7	Favorites_count	Number of tweets liked.	Available	23	profile_sidebar_fill_color	Color of the sidebar background.	Deprecated

8	listed_count	Number of times user is listed.	Available	24	profile_background_image T_huerl	HTTPS URL of the background image.	Deprecated
9	url	Profile URL	Available	25	profile_background_color	Hex code for profile background color.	Deprecated
10	Lang	Main language of tweets.	Deprecated	26	profile_link_color	Hex code for profile link color.	Deprecated
11	time_zone	User's time zone.	Deprecated	27	utc_offset	Time offset (in seconds) from GMT.	Available
12	Location	User's location.	Available	28	Protected	Indicates if the account is private (binary).	Available
13	default_profile	Indicates if default profile theme is used.	Available	29	Verified	Indicates if the account is verified.	Deprecated
14	default_profile_ image	Indicates if the default profile image is used.	Available	30	Notifications	Indicates if user receives status alerts.	Available
15	geo_enabled	Indicates whether location sharing is enabled.	Deprecated	31	Description	User-provided account description (UTF-8 text)..	Available
16	profile_image_u rl	URL of the user's profile image.	Deprecated				

5 Proposed Bot Detection System Design

The proposed system is designed to detect real-time XCorp bots that promote specific hashtags. It incorporates both user-based and content-based features, including the polarity and subjectivity of tweets, to accurately identify automated accounts. These features were chosen because user-based metrics capture account behavior, while content-based features allow analysis of the textual patterns that bots typically produce. The following section presents the system architecture, data processing workflow, and feature extraction methodology.

In terms of system architecture, the system is divided into two main phases: the first phase involves building the proposed bot detection system, and the second phase involves applying the proposed bot detection system on tweets in real time. As shown in Figure 1, the proposed bot detection system first collects tweets from the dataset. The tweets are then preprocessed using filtering to remove non-alphanumeric characters, tokenization using the NLTK word_tokenize function, stop word removal using the standard NLTK English stop word list, and stemming using the NLTK PorterStemmer. These preprocessing steps were selected to standardize the text, reduce noise, and improve the accuracy of sentiment analysis, which is particularly sensitive to word variations and extraneous characters. After preprocessing, the tweets are analyzed with the TextBlob library to calculate sentiment scores based on polarity and subjectivity, as it provides a simple yet effective lexicon-based approach suitable for real-time processing. TextBlob was selected for its simplicity, interpretability, and efficiency, allowing sentiment scores to be seamlessly integrated into the Spark-based bot detection pipeline while balancing accuracy and computational performance.

Afterward, the sentiment outputs, which consist of polarity and subjectivity scores, are merged with the other independent features provided by the dataset. A correlation-based feature selection method is then applied to evaluate the relationship between each independent feature and the dependent variable (bot/human). Specifically, Pearson correlation analysis is used to measure the linear association between features and the target label. This method is selected due to its simplicity, interpretability, and efficiency in identifying irrelevant features. Positive correlation values indicate that both variables increase together, while negative values indicate an inverse relationship. Values close to zero suggest little or no linear association. Since Pearson correlation requires numerical inputs, categorical features are first converted into numerical form using the StringIndexer method in PySpark. After computing the correlation scores, features with zero correlation values are removed, as they do not contribute to the prediction task, while the remaining features are retained for training the machine learning models. Although Pearson correlation evaluates features individually and assumes linear relationships, it is used in this study as a preliminary filtering step to remove clearly irrelevant features. The machine learning models employed in this study, such as RF and Gradient Boosted Trees (GBT), are then responsible for capturing complex feature interactions and non-linear patterns.

After PySpark preprocessing, the data is divided into training and test sets and used by five PySpark MLlib classification algorithms to build models. These five models were chosen, GBT, Linear SVC, DT, LR, and RF, because they cover a diverse range of learning strategies (ensemble, linear, and tree-based), which ensures diverse learning approaches and robust model evaluation. The model with the highest accuracy and F-measure is selected as the final model.

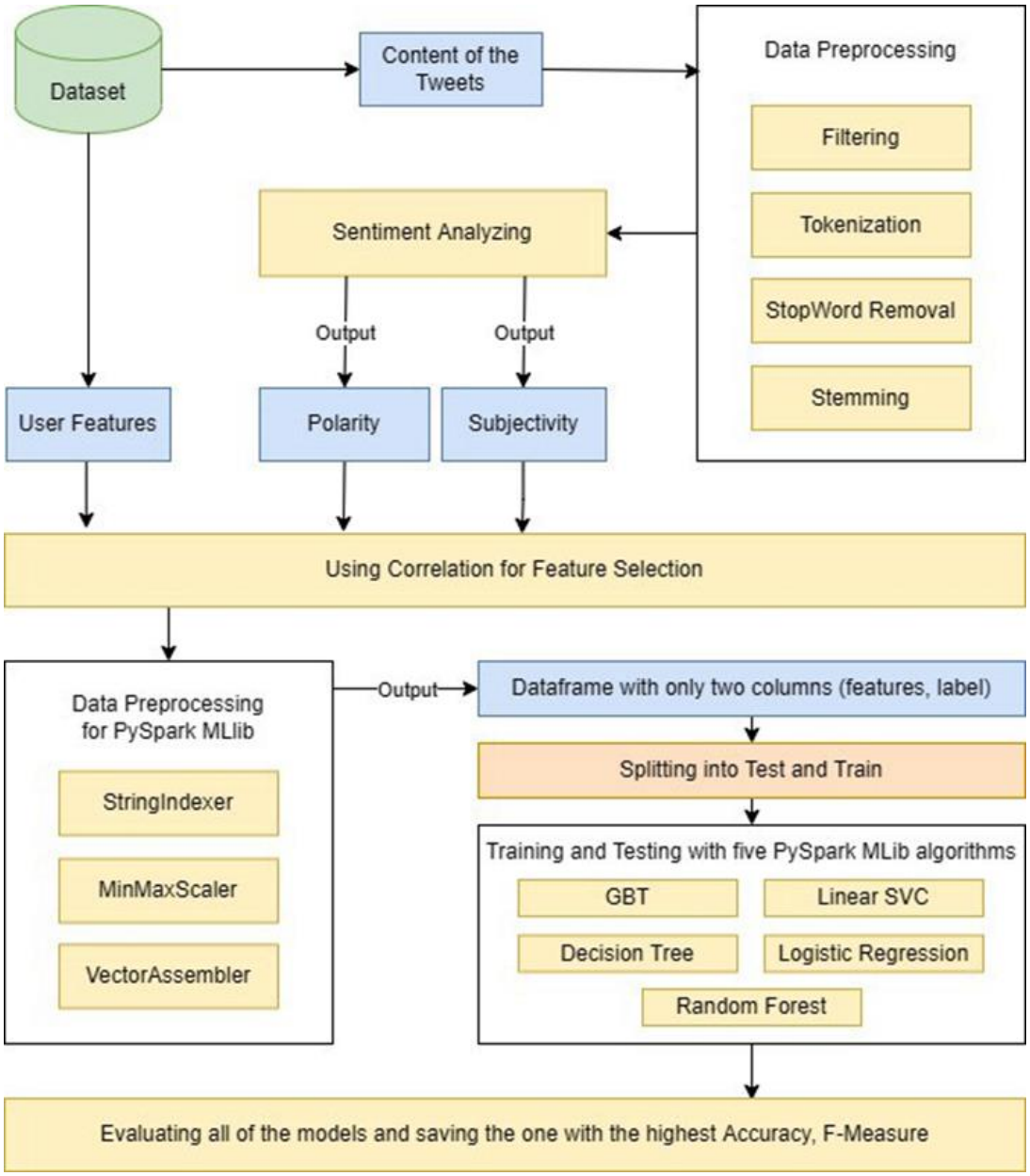

Figure 1: Workflow of the Proposed Bot Detection System.

The second phase of the system involves applying the proposed bot detection system to real-time tweets. This phase begins with streaming data from the XCorp platform, followed by detecting automated accounts using both user-based and content-based features, including sentiment scores derived from tweet text. As shown in Figure 2, the system prompts the user to input a hashtag. Based on the input, tweets containing the hashtag are collected. The same preprocessing and sentiment calculation methods from phase one are applied. Simultaneously, user-based features are collected, and the average sentiment scores are merged with each user's features. The combined features are processed using StringIndexer, MinMaxScaler, and VectorAssembler to create a DataFrame suitable for the saved model. The model then predicts whether each user is a bot or human, stores the results in a CSV file, and loops back to process new users in real time.

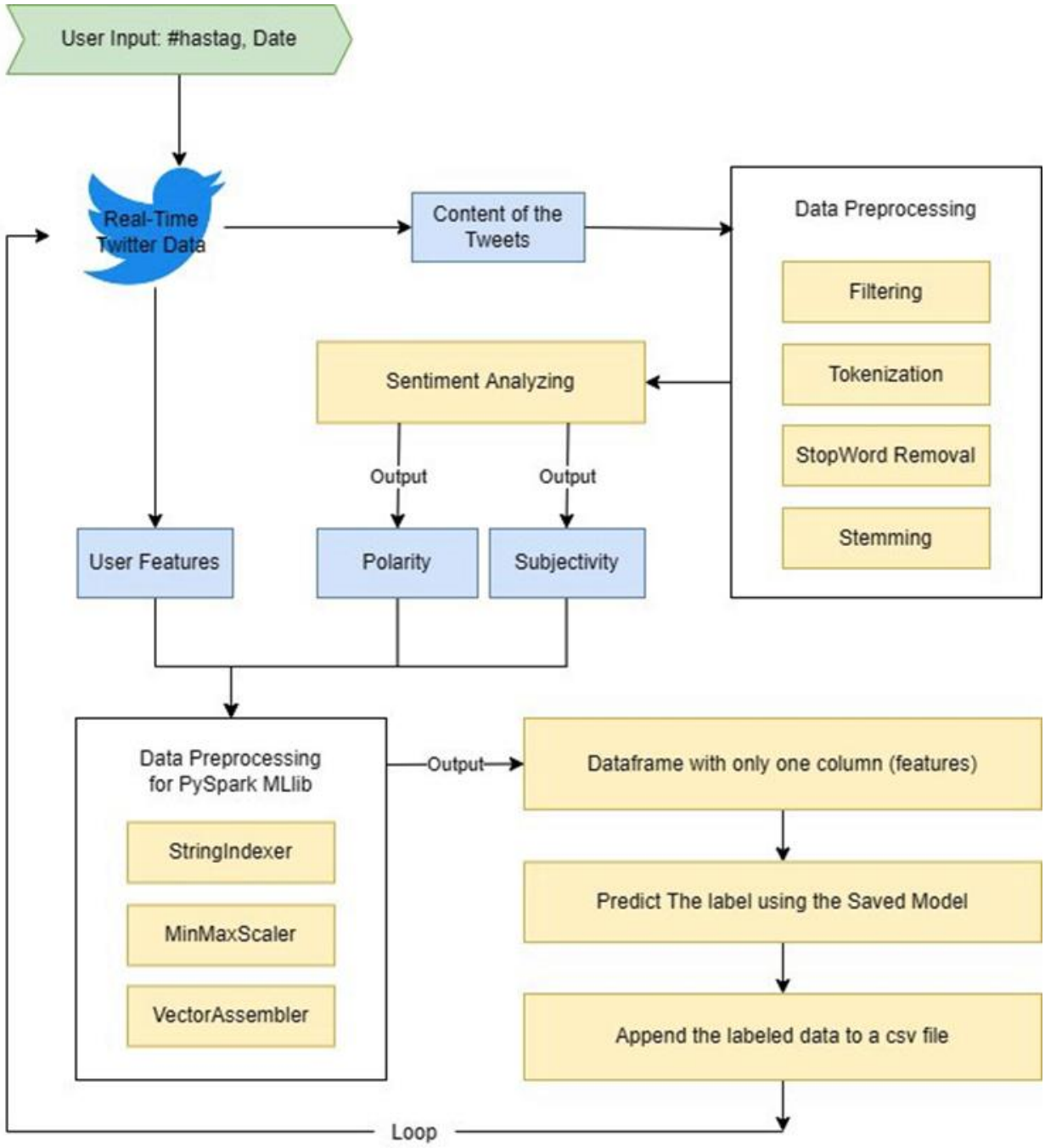

Figure 2: Real-Time Tweet Analysis Using the Proposed Bot Detection System.

The pseudocode in Figure 3 outlines the process for calculating average sentiment scores for each account. First, all tweets and account data are loaded into separate DataFrames. Then, for each account, the system iterates through all tweets posted by that account. For each tweet, the polarity and subjectivity scores are calculated using the TextBlob library. After processing all tweets for an account, the average polarity and subjectivity scores are computed. These average values are then appended to the corresponding account in the account DataFrame. This procedure generates content-based features that are merged with user-based features and subsequently used as input for the machine learning models.

Pseudocode: Calculation and Addition of Average Polarity and Subjectivity Scores to User Accounts

Input: Tweets DataFrame & Accounts DataFrame

Output: Updated Accounts DataFrame with average polarity and subjectivity scores

Begin:

1. Load Tweets DataFrame and Accounts DataFrame
2. **For each Account in Accounts DataFrame**:
3. Initialize total_polarity = 0, total_subjectivity = 0
4. Initialize tweet_count = 0
5. **For each Tweet posted by the Account:**
6. Calculate the tweet's polarity score using TextBlob
7. Calculate the tweet's subjectivity score using TextBlob
8. Increment total_polarity and total_subjectivity
9. Increment tweet_count
10. **End For**
11. Compute average_polarity = total_polarity / tweet_count
12. Compute average_subjectivity = total_subjectivity / tweet_count
13. Append average_polarity and average_subjectivity to the Account record.
14. **End For**
15. **Return: Updated Accounts DataFrame**

End

Figure 3: Pseudocode for Calculating and Adding Average Polarity and Subjectivity Scores to User Accounts.

6 Implementation, Results, and Discussion

The proposed system was developed and executed in a computing environment optimized for real-time and large-scale data processing. Experiments were conducted on a machine with an Intel® Core™ i7-7500U CPU at 2.70 GHz, running Debian GNU/Linux 11 (Bullseye) as the operating system to ensure stability and compatibility with Apache Spark. The system utilized PySpark version 3.1.2 for distributed data processing and Python for feature extraction and model training. Real-time data from XCorp was accessed via the XCorp API, which required creating a developer account and authenticating with API keys. This setup ensured secure and continuous access to streaming data, which is essential for bot detection in real-time environments. The performance of the proposed system was evaluated in terms of accuracy, precision, recall, F1-score, and CPU time. This evaluation considered models with and without content-based features across all five machine learning methods: GBT, Linear SVC, DT, LR, and RF.

6. 1 Implementation and Results Using User-Based Features

This model was constructed using only user-based features. In this scenario, only the user-related dataset was considered. The features including Profile-Banner-URL, Description, Location, Profile-Image-URL-HTTPS, Status-Count, URL, Favorite-Count, Default-Profile, Listed-Count, Friends-Count, Created-At, Verified, Followers-Count, and Default-Profile-Image were preprocessed using **StringIndexer**, **MinMaxScaler**, and **VectorAssembler** functions.

To ensure a robust assessment, we applied an 80–20 stratified train–test split, reserving 20% of the data for independent testing, and performed 10-fold cross-validation on the training set to reduce variance. The processed data was then used to train and evaluate five PySpark ML algorithms: GBT, Linear SVC, DT, LR, and RF. Performance metrics including accuracy, precision, recall, F1-score, and CPU time were computed to provide an unbiased evaluation of model generalization. The results are summarized in Table 3, while computational time in milliseconds is reported in Table 4.

Table 3: Accuracy, Recall, Precision, and F1-Score Using User-Based Features.

Algorithm	Accuracy	Recall	Precision	F1-Score
GBT Classification	97.3%	98.0%	94.6%	**96.27%**
Linear SVC Classification	97.8%	98.8%	93.6%	**96.13%**
Decision Tree Classification	98.5%	98.8%	95.6%	**97.17%**
Logistic Regression	97.1%	98.7%	90.7%	**94.53%**
Random Forest Classification	98.4%	98.7%	97.0%	**97.84%**

Table 4: CPU Time Using User-Based Features.

Algorithm	Time in Millisecond
GBT Classification	35.5
Linear SVC Classification	94.5
Decision Tree Classification	13.1
Logistic Regression	30.2
Random Forest Classification	15.7

6. 2 Implementation and Results Using Both User- and Content-Based Features

The proposed model was developed using a combination of user- and content-based features. The tweets dataset was first preprocessed through filtering, tokenization, stop-word removal, and stemming to ensure consistency and reduce noise. Sentiment analysis was then performed using **TextBlob**, calculating polarity and subjectivity scores for each tweet. These scores were averaged per user and merged with sixteen user-based features: Profile-Banner-URL, Description, Location, Profile-Image-URL-HTTPS, Status-Count, URL, Favorite-Count, Default-Profile, Listed-Count, Friends-Count, Created-At, Verified, Followers-Count, and Default-Profile-Image. All features were subsequently preprocessed using **StringIndexer**, **MinMaxScaler**, and **VectorAssembler** to produce a feature matrix suitable for Spark ML classification.

For the model using both user- and content-based features, the dataset was processed and split into training and testing sets following the same 80–20 stratified split and 10-fold cross-validation procedure described in Section 6.1. The combined feature matrix was then used to train and evaluate the same five PySpark ML algorithms: GBT, Linear SVC, DT, LR, and RF. Performance metrics including Accuracy, Recall, Precision, F1-score, and CPU time were computed. The results are summarized in Table 5, with computational time in milliseconds reported in Table 6.

Table 5: Accuracy, Recall, Precision, and F1-Score Using Both User- and Content-Based Features.

Algorithm	Accuracy	Recall	Precision	F1-Score
GBT Classification	98.9%	99.8%	95.6%	**97.65%**
Linear SVC Classification	98.8%	99.8%	94.3%	**96.97%**
Decision Tree Classification	98.8%	99.7%	97.0%	**98.33%**
Logistic Regression	97.6%	99.2%	92.2%	**95.57%**
Random Forest Classification	99.6%	99.7%	99.1%	**99.40%**

Table 6: CPU Time Using Both User- and Content-Based Features.

Algorithm	Time in Millisecond
GBT Classification	64.4
Linear SVC Classification	77.2
Decision Tree Classification	14.8
Logistic Regression	32.9
Random Forest Classification	14.9

6. 3 Discussion and Analysis of Model Performance

This section discusses the findings reported in Section 6. The analysis is organized into five subsections, focusing on Accuracy, Recall, Precision, F1-Score, and CPU time.

1. **Accuracy:** As shown in Figure 4, adding content-based features improves the accuracy of all five algorithms. For GBT Classification, the model using only user-based features achieves 97.3% accuracy, which increases by 1.6% to 98.9% when content-based features are included. Similarly, the other four classifiers also benefit from content-based features. Among them, Random Forest Classification achieves the highest accuracy of 99.6%.

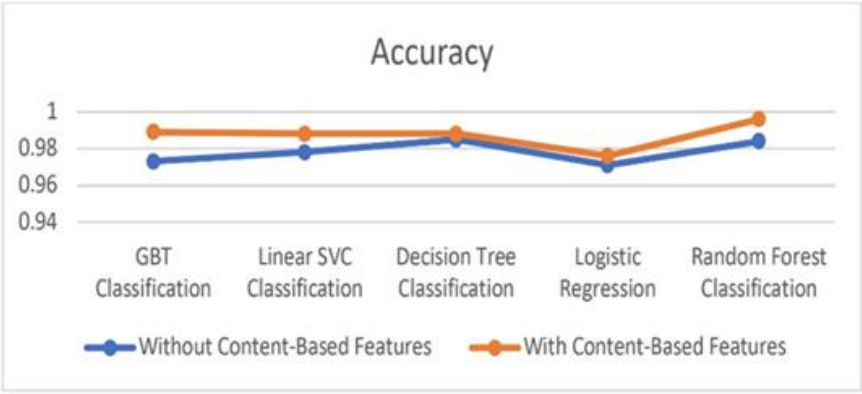

Figure 4: Accuracy Comparison Between Models With and Without Content-Based Features.

2. **Recall:** As shown in Figure 5, the addition of content-based features increases the recall scores for all five algorithms. With GBT Classification, the recall of a model generated using only user-based features is 98.0%; however, when content-based features are incorporated, the recall increases by 0.018, reaching 99.8%. Similar to the accuracy results, the inclusion of content-based features improves the recall of the other four classification algorithms as well. Random Forest Classification again achieves high recall score, reaching 99.7% with content-based features compared to 98.7% without them.

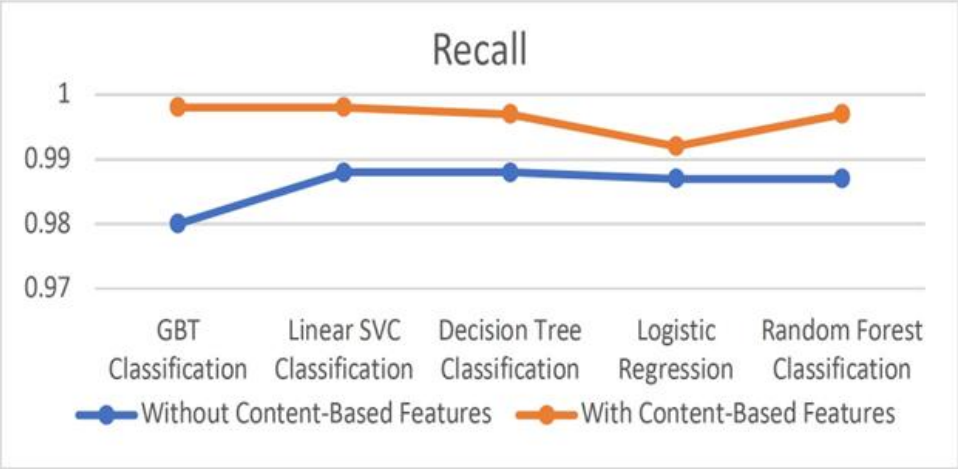

Figure 5: Recall Comparison Between Models with and Without Content-Based Features.

3. **Precision:** Similar to accuracy and recall, the addition of content-based features increases the precision scores for all algorithms. With GBT Classification, the precision of a model built without content-based features is **94.6%**; however, when content-based features are incorporated, the precision increases to **95.6%**. Likewise, the inclusion of content-based features improves the precision of the other four classification algorithms. Nevertheless, Random Forest Classification achieves the highest precision score, reaching **99.1%** with content-based features compared to **97.0%** without them. Figure 6 presents the precision score comparison chart.

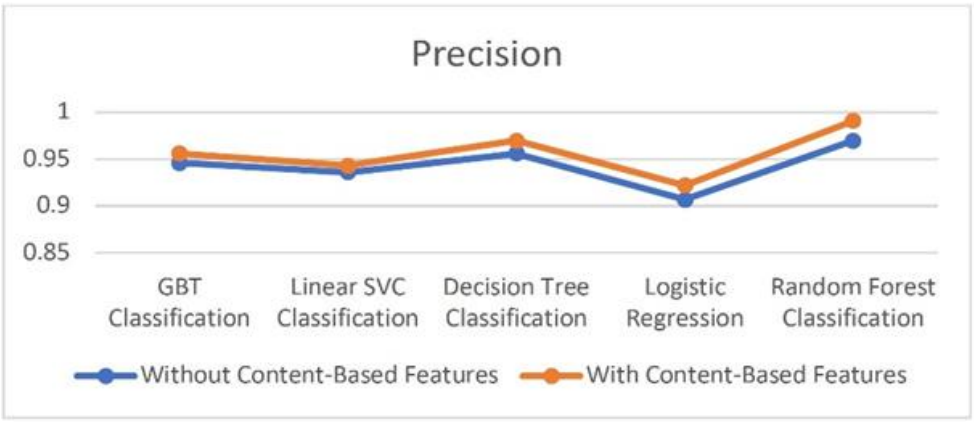

Figure 6: Precision Comparison Between Models with and Without Content-Based Features.

4. **F1-score:** In addition to accuracy, recall, and precision, the F1-score was calculated to provide a balanced evaluation of the models. As shown in Tables 3 and 5 , Random Forest achieved the highest F1-score in both scenarios, reaching 97.84% before adding content-based features and 99.40% after incorporating polarity and subjectivity, confirming its superior ability to balance precision and recall in detecting bot accounts. Figure 7 presents the F1-score comparison chart.

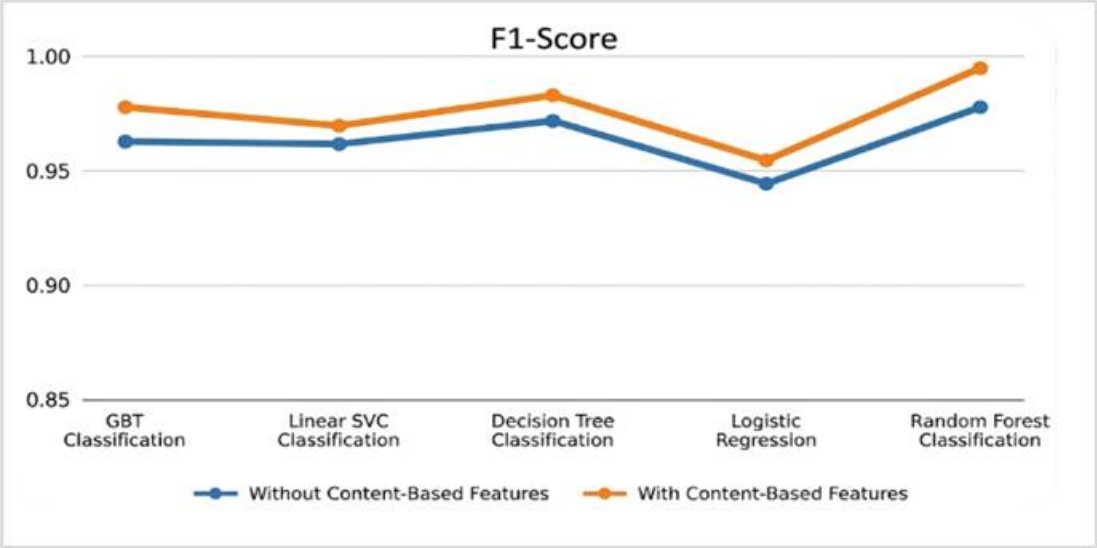

Figure 7: F1-score Comparison Between Models with and Without Content-Based Features.

5. **CPU-Time:** As illustrated in Figure 8, adding content-based features does not significantly affect CPU time. In particular, when using Random Forest and Decision Tree Classifications, the difference is negligible. Decision Tree Classification requires 14.8 milliseconds with content-based features and 13.1 milliseconds without them, while Random Forest Classification requires 14.9 milliseconds with content-based features and 15.7 milliseconds without them.

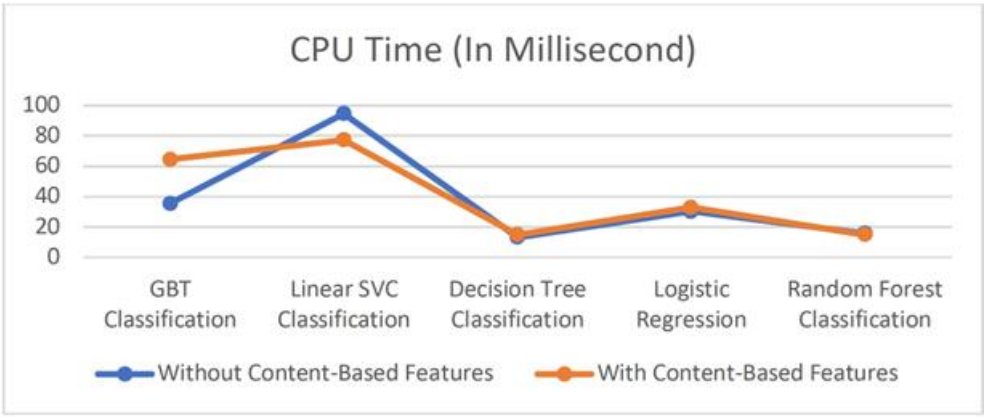

Figure 8: CPU-Time Comparison Between Models with and Without Content-Based Features.

Considering the accuracy, recall, precision, F1-score, and CPU time, it can be concluded that adding content-based features significantly improves the model performance. Specifically, the accuracy, recall, precision, and F1-score values increase when content-based features are incorporated, while the computational time remains relatively constant. Furthermore, based on these evaluation metrics, Random Forest Classification provides the best performing model among the algorithms used in this study.

6. 4 Comparison with Existing Methods

As discussed in Section II, numerous studies have explored spam bot detection and user classification on XCorp using different features and methodologies. Wang employed graph-based and tweet-based features with various classifiers, achieving 91% accuracy using the Naïve Bayes model. Zi Chu et al. classified users into human, bot, and cyborg categories, reporting an average accuracy of 96.0%. Gilani et al. proposed a systematic approach based on user behavior, achieving 86.44% accuracy, while Pozzana et al. focused on tweet-level metrics and obtained approximately 86% accuracy. Monica and Nagarathna utilized sentiment-based analysis and reported an accuracy of 97%, whereas Anisha et al. addressed spam classification and sentiment analysis separately, achieving 97.78% and 70.56% accuracy, respectively. Although these studies report strong performance, it is important to note that they were evaluated on different datasets and experimental settings, making direct comparison not strictly equivalent. Nevertheless, the proposed approach in this study integrates user-based features with sentiment analysis, leading to improved performance under the evaluated conditions. Figure 9 presents a comparative overview

of these studies and the proposed system in terms of accuracy, precision, and recall.

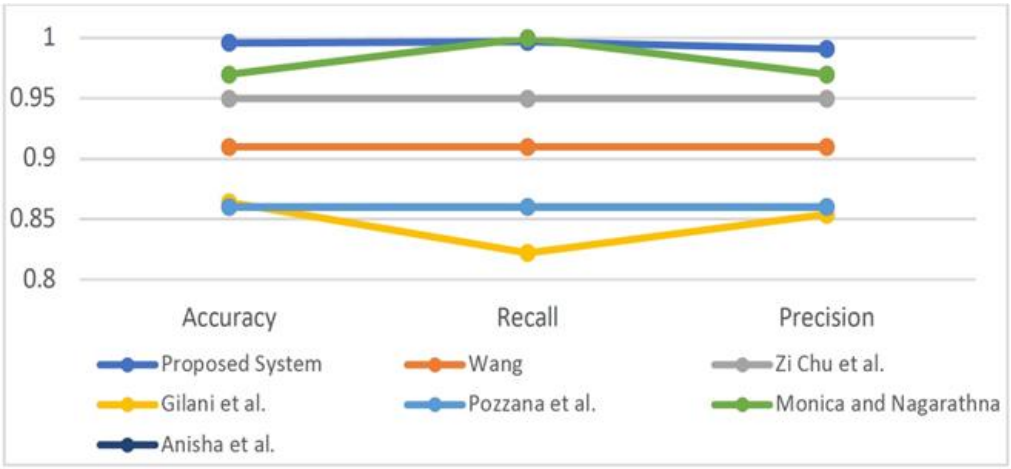

Figure 9: Performance Evaluation of the Proposed Model vs. Existing Approaches.

7 Conclusions and Future Work

Based on the suggested system architecture and the findings presented in this study, several key conclusions can be drawn. The main objective of this study was to identify XCorp bots involved in promoting a specific hashtag using real-time data. To achieve this objective, a comprehensive system was proposed consisting of several phases, including data collection, preprocessing, sentiment analysis, feature extraction, and classification. In addition to XCorp user features, content-based features such as polarity and subjectivity were incorporated into the model, which contributed to improving the overall performance of the system. In total, fourteen user-based features were used together with polarity and subjectivity scores derived using TextBlob functions to perform sentiment analysis on tweet contents. Apache Spark machine learning techniques were employed to accelerate the implementation of the system and handle large-scale data efficiently. Five machine learning algorithms, namely GBT Classification, Linear SVC Classification, Decision Tree Classification, Logistic Regression, and Random Forest Classification, were used to construct and evaluate the models. The evaluation results of the models with and without content-based features showed that incorporating these features significantly improved accuracy, recall, and precision scores, while the computational time remained relatively constant. Among the evaluated algorithms, Random Forest Classification produced the best results, achieving an accuracy of 99.6%, recall of 99.7%, and precision of 99.6%. In addition, the proposed system enables real-time prediction using data directly streamed from the XCorp platform.

For future work, additional content-based features such as average posting time and tweet length could be incorporated along with polarity and subjectivity scores to improve the model's ability to identify XCorp bots promoting specific hashtags. Future studies could also explore advanced sentiment analysis techniques, including transformer-based models, to enhance performance. The approach could be extended to a web-based platform for real-time bot detection, content authenticity analysis, fraud detection, and rumor identification, helping mitigate the spread of misinformation.

References

[1] Kwak, H., Lee, C., Park, H. & Moon, S. What is Twitter, a social network or a news media?. *In Proceedings of the 19th international conference on World wide web (ACM, New York, NY, USA*), doi: 10.1145/1772690.1772751 (2010).

[2] Shahi, G. K., Dirkson, A. & Majchrzak, T. A. An exploratory study of COVID-19 misinformation on Twitter. *Online Soc. Netw. Media* **22**, 100-104, doi: 10.1016/j.osnem.2020.100104 (2021).

[3] Rasul, H. M. & Jumaa, A. K. Real-time Twitter data analysis: A survey. *UHD J. Sci. Technol*. **6**, 147–155, doi:10.21928/uhdjst.v6n2y2022.pp147-155 (2022).

[4] Omar, H. K. & Jumaa, A. K. Distributed big data analysis using spark parallel data processing. *Bull. Electr. Eng. Inform*. **11**, 1505–1515, doi: 10.11591/eei.v11i3.3187 (2022).

[5] Esmaeilzadeh, A., Heidari, M., Abdolazimi, R., Hajibabaee, P. & Malekzadeh, M. Efficient large scale NLP feature engineering with Apache spark. *In IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, doi: 10.1109/CCWC54503.2022.9720765 (2022).

[6] Mohammed, S. S., Mohammed, H. I., Mohammed, N. S., Shafiq, N. M. & Flayyih, H. Q. Future Directions in Artificial Intelligence for Cybersecurity: Emerging Trends and Key Developments. *Passer J. Basic Appl. Sci.*, doi:10.24271/psr.2025.521050.2126 (2026).

[7] Twitter Developer Documentation. "Getting started with the Twitter API." Available: https://developer.twitter.com/en/docs/twitter-api/getting-started/getting-access-to-the-twitter-api, Accessed: Mar. 15, 2026.

[8] Wang, A. H. Detecting spam bots in online social networking sites: A machine learning approach. *In Proc. 24th Annual IFIP WG 11.3 Working Conf. on Data and Applications Security and*

Privacy, 335–342 (Rome, Italy), doi: 10.1007/978-3-642-13739-6_25 (2010).

[9] Chu, Z., Gianvecchio, S., Wang, H. & Jajodia, S. Detecting automation of twitter accounts: Are you a human, bot, or cyborg?. *IEEE Trans. Dependable Secure Comput.* **9**, 811–824, doi: 10.1109/TDSC.2012.75 (2012).

[10] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A. & Tesconi, M. Fame for sale: Efficient detection of fake Twitter followers. *Decis. Support Syst.* **80**, 56–71, doi: 10.1016/j.dss.2015.09.003 (2015).

[11] Gilani, Z., Kochmar, E. & Crowcroft, J. Classification of twitter accounts into automated agents and human users. *In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (ACM, New York, NY, USA), 489–496, doi: 10.1145/3110025.3110091 (2017)

[12] Cresci, S., Pietro, R. D., Petrocchi, M., Spognardi, A. & Tesconi, M. The paradigm shift of social spambots: Evidence, theories, and tools for the arms race. *In Proc. 26th Int. Conf. World Wide Web Companion* (WWW '17 Companion, Perth, Australia), 963–972, doi: 10.1145/3041021.305513514 (2017).

[13] Pozzana, I. & Ferrara, E. Measuring bot and human behavioral dynamics. *Front. Phys*. **8**, doi: 10.3389/fphy.2020.00125 (2020).

[14] Monica, C. & Nagarathna, N. Detection of fake tweets using sentiment analysis. *SN Comput. Sci* **1**, doi: 10.1007/s42979-020-0110-0 (2020)

[15] Rodrigues, A. Fernandes, P. Aakash, A. Abhishek, A. Shetty, A. Atul, K. Lakshmanna, K. & Shafi, R. M. Real-time Twitter spam detection and sentiment analysis using machine learning and deep learning techniques. *Comput. Intell. Neurosci* **2022**, 1–14, doi: 10.1155/2022/5211949 (2022).

[16] Abdulhammed, O. & Karim, P. Sentiment analysis using SVM-based SSO intelligence algorithm. *Passer J. Basic Appl. Sci.* **4**, 17–39, doi: 10.24271/psr.2022.160801 (2022).

[17] Lei, Z. Wan, H. Zhang, W. Feng, S. Chen, Z. Li, J. Zheng, Q. & Luo, M. BIC: Twitter Bot Detection with Text Graph Interaction and Semantic Consistency. *In Proc. 61st Annual Meeting of the Association for Computational Linguistics (ACL)* (Toronto, Canada), 10326–10340, doi: 10.18653/v1/2023.acl-long.575 (2023)

[18] Wei, C., Liang, G. & Yan, K. BotGSL: Twitter bot detection with graph structure learning. *Comput. J.* **67**, 2486–2497, doi: 10.1093/comjnl/bxae020 (2024).

[19] Anshul, A., Rehman, M. Z. U., Kadali, S. A. & Kumar, N. RoGBot: Relationship-oblivious graph-based neural network with contextual knowledge for bot detection. *arXiv:2510.23648,* doi: 10.48550/ARXIV.2510.23648 (2025).

[20] Luo, J. & Jin, C. Fusing content and social relationships: A multi modal heterogeneous graph transformer approach for social bot detection. *EPJ Data Sci* **14**, doi: 10.1140/epjds/s13688-025-00583-5 (2025).

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

[22] Messaoudi, C., Guessoum, Z. & Ben Romdhane, L. Opinion mining in online social media: a survey. *Soc. Netw. Anal. Min.* **12**, doi: 10.1007/s13278-021-00855-8 (2022).

[23] Hayder, W. Supervised sentiment analysis model of textual content for images. *Passer J. Basic Appl. Sci.* **2**, doi: 10.24271/psr.16 (2020)

[24] TextBlob: Simplified Text Processing, version 0.16.0, 2020. [Online]. Available: https://textblob.readthedocs.io/en/latest/ (accessed Mar. 2026).

[25] Singh, P. Manage Data with PySpark. in Machine Learning with PySpark: With Natural Language Processing and Recommender Systems. *Processing and Recommender Systems*, *Berkeley, CA, USA: Apress*, 15–37, doi: 10.1007/978-1-4842-7777-5_2 (2022).

[26] "MIB Dataset: Italian Twitter Bot Repository," [Online]. Available: http://mib.projects.iit.cnr.it/dataset.html. [Accessed: Mar. 25, 2026].