Jurnal ELTIKOM :
Jurnal Teknik Elektro, Teknologi Informasi dan Komputer

Vol. 9, No. 2, December 2025, page. 206-217
ISSN 2598-3245 (Print), ISSN 2598-3288 (Online)
DOI: http://doi.org/10.31961/eltikom.v9i2.1727
Available online at http://eltikom.poliban.ac.id

OPTIMIZATION MODEL FOR FAKE ACCOUNT DETECTION ON TWITTER (X) SOCIAL MEDIA USING FEATURE ENGINEERING AND MACHINE LEARNING APPROACHES

Ni Nyoman Eny Perimawati*, Roy Rudolf Huizen, Dandy Pramana Hostiadi
Department of Magister Information System, Institut Teknologi dan Bisnis STIKOM Bali, Indonesia
e-mail: 232012011@stikom-bali.ac.id, roy@stikom-bali.ac.id, dandy@stikom-bali.ac.id

Received: 6 July 2025 – Revised: 1 September 2025 – Accepted: 15 October 2025

ABSTRACT

Twitter (X) has become an important platform for community interaction, but this also creates serious challenges due to the proliferation of fake accounts that can harm users and undermine credibility. Previous studies have proposed detection methods but often lacked forensic analysis based on extracted feature information. This study utilizes labeled datasets and supervised evaluation metrics (precision, recall, and F1-score) to validate model performance. Extracting behavioral information from features is crucial for achieving accurate and reliable detection results. The study introduces a novelty in the form of engineered behavioral features that significantly enhance detection accuracy, achieving up to 99.94% using AdaBoost. The proposed approach detects fake accounts on Twitter (X) by extracting key feature information and developing an optimal detection method through machine learning algorithms, including Random Forest, SVM, and AdaBoost. Furthermore, the model is optimized using feature engineering techniques. The novelty of this work lies in the development of engineered features through distribution analysis based on data characteristics and the improvement of classification performance through feature engineering optimization. The initial experiment without feature engineering shows that Random Forest achieved the highest accuracy of 98.77%, followed by AdaBoost at 98.57% and SVM at 95.90%. After applying feature engineering, performance improved, with AdaBoost reaching 99.94%, Random Forest 99.69%, and SVM 99.32%. The proposed model can assist system analysts in detecting fake accounts and contribute to solving forensic cybercrime challenges, particularly in identifying fake social media profiles.

Keywords*: fake accounts, feature engineering, machine learning, twitter (X).*

I. INTRODUCTION

SOCIAL media has become an inseparable part of people's daily lives. The most frequently used platforms include Facebook, Instagram, TikTok, and Twitter (X). Millions of users worldwide use Twitter (X) to communicate, share information, and build communities. However, as social media becomes increasingly popular, a new problem has emerged: the proliferation of fake accounts that can harm and mislead users. Research shows that more than 15% of social media accounts can be considered fake or bot accounts [1]. Such accounts can undermine the credibility and security of both platforms and users, especially when they are used for activities that violate policies or laws, such as spreading false information or conducting social attacks [2].

Fake accounts often employ sophisticated techniques to mimic the behavior of real users, making manual detection difficult [3]. They are typically created to deceive or conceal the true identity of their users. Several types of accounts can be classified as fake accounts, including those created by humans with malicious intent to deceive others or spread false information (hoaxes), those created and managed by bots for automated purposes that frequently disseminate spam or misinformation, and cyborgs, which are hybrid accounts combining human and bot activity. In this study, fake accounts refer specifically to those automatically created by bots (machines) used for malicious activities such as spreading hoaxes, spam, and unethical content on social media platforms [4].

While clustering-based approaches such as PSO-KMeans have been investigated for fake account detection [5],[6], this study takes a different approach by validating supervised machine learning using labeled data. The study benchmarks against datasets such as the Cresci Twitter Bot Repository [6], ensuring alignment with current studies. The contribution lies in demonstrating that engineered behavioral features, including interaction ratios, posting density, and account age, can significantly enhance the performance of supervised classification models.

Several studies have shown that fake accounts created by bots display behavioral patterns different from those of genuine accounts created by humans, particularly in activity frequency, timing, interaction patterns, and types of shared content. Detecting fake accounts through machine learning has become a popular method because of its ability to identify complex patterns in user behavior data [7]. Previous research demonstrated that the Decision Tree algorithm can effectively identify hoax-related news often associated with fake accounts [8]. Similarly, Yoan Maria Vianny and Erwin Budi Setiawan found that the J48 algorithm can detect rumors frequently spread by fake accounts and emphasized the importance of feature analysis in identifying suspicious accounts [9]. Fake accounts often show abnormal activities such as sudden spikes in followers, automated tweets, or irregular activity patterns that are uncommon among real users [10]. Analyzing such behavioral characteristics remains one of the most promising approaches for detecting fake accounts on Twitter.

The machine learning approach can be applied to detect fake accounts, as its algorithms can identify common patterns and characteristics associated with such accounts. This method has gained popularity because of its ability to recognize complex patterns in user behavior data [11]. Machine learning algorithms can analyze various behavioral features, including posting frequency, activity time, and interaction patterns with other users. Feature engineering plays a crucial role in enhancing detection accuracy since carefully selected features can significantly improve predictive performance. Therefore, selecting and optimizing the most informative behavioral features is essential. Effective feature engineering allows machine learning models to identify fake accounts more accurately by utilizing variables such as the number of followers, retweets, likes, and posting frequency. The choice of relevant features is a key factor influencing the performance of classification models based on machine learning [12]. Selecting and processing appropriate features can greatly improve the model's effectiveness in detecting fake accounts [13].

This study employs a supervised machine learning model for classification, as previous research has demonstrated its strong performance in bot detection. The attributes used to identify fake accounts include account identity variables such as name, location, profile picture, number of friends, language, and number of followers, without considering behavioral data [14]. In addition, feature engineering was conducted to calculate several parameters: the number of activities for each tweet to analyze posting patterns, the number of tweets per user to indicate posting activity, and the average number of replies, retweets, and quotes per user to show the level of interaction. Additional features include the number of unique locations used in tweets, as fake accounts often reuse the same location, the number of images attached to tweets, the average number of likes received, and account age.

This study focuses on detecting fake accounts on the Twitter platform through feature engineering to generate new attributes that improve the performance of machine learning algorithms, specifically Random Forest, SVM, and AdaBoost. The model's performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and the confusion matrix. The study aims to develop a detection method using Random Forest, SVM, and AdaBoost with a feature engineering approach. Furthermore, the model is evaluated to determine the most effective algorithm for detecting fake accounts. The findings of this study are expected to serve as a reference for future research on similar topics. The novelty of this study lies in the development of engineered features through distribution analysis based on data characteristics in a fake account detection case study and the enhancement of classification performance through feature engineering optimization.

II. RESEARCH METHOD

This study proposes an optimized model for detecting fake accounts on Twitter (X) using a feature engineering approach. The detection process applies machine learning-based classification methods.

TABLE 1
FEATURE DECRYPTION ON TWITTER (X)

No	Feature Name	Feature Description
1	conversation_id_str	unique identification for conversation sessions
2	created_at	Date and time when the tweet was created
3	favorite_count	Number of favorites received by the tweet
4	full_text	Full text content of the tweet
5	id_str	Unique ID for the tweet
6	image_url	URL of the image associated with the tweet
7	in_reply_to_screen_name	Username of the account being replied to
8	lang	Language used in the tweet
9	location	User's location when creating the tweet
10	quote_count	Number of quotes received by the tweet
11	reply_count	Number of replies received by the tweet
12	retweet_count	Number of retweets received by the tweet
13	tweet_url	URL link of the tweet
14	user_id_str	Unique ID for the user
15	username	Username of the account that created the tweet

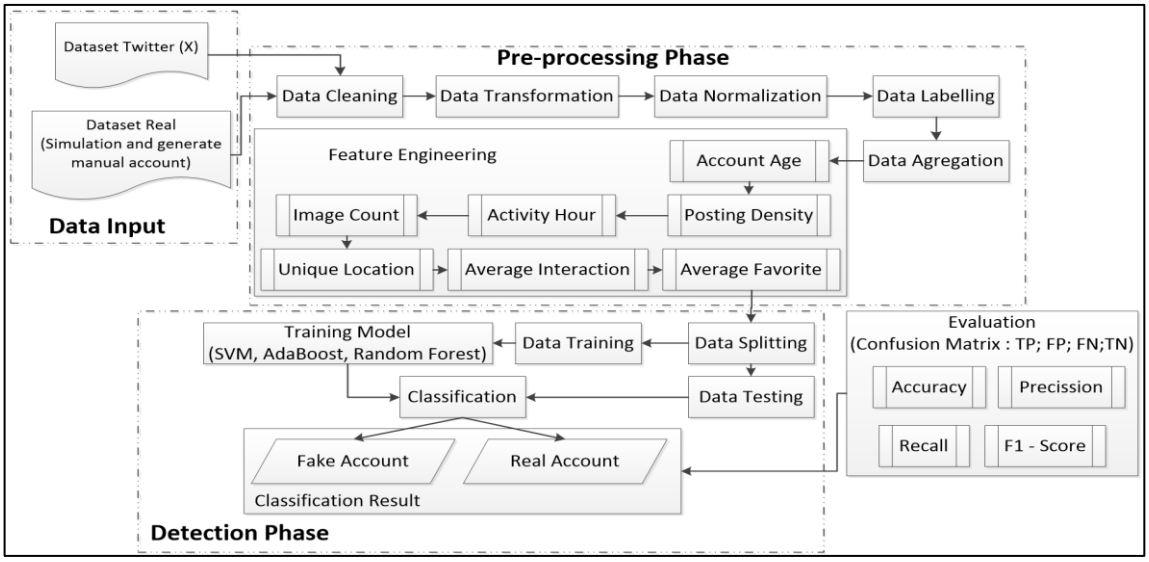


Figure 1. Proposed Model Overview

The proposed model is illustrated in Figure 1.

A. Preprocessing

This study uses two types of data. The first dataset was collected by crawling Twitter (X) data through the Twitter API, while the second dataset was obtained through manual simulation and the creation of new accounts. The description of the features extracted from the crawling process is presented in Table 1.

The initial stage of this study involves a preprocessing phase, in which the model performs data cleaning, transformation, normalization, labeling, aggregation, and feature engineering. In the data cleaning stage, duplicate entries are removed based on the following criteria: records with identical *username* and *user_id_str* values, identical *full_text*, and identical *conversation_id_str* and timestamp (*created_at*) attributes. Duplicate removal is conducted using Python with the *drop_duplicates* method based on these criteria.

The data transformation stage converts raw data into a structured format. For instance, the *created_at* feature is converted from a string to a date data type, while *favorite_count*, *reply_count*, and *retweet_count* are converted from strings to numeric data types. Data normalization follows, standardizing feature values within a specific range.

Next, the labeling process assigns values of 0 to posts from fake accounts and 1 to posts from real accounts. Expert analysis is used to validate the authenticity of both fake and genuine accounts. The labeled dataset combines 812 real accounts obtained through data crawling with 100 simulated fake accounts, which were validated by expert review. Feature engineering extends beyond basic metadata

TABLE 2
DETAILED DESCRIPTION FOR FEATURE ENGINEERING

No	Original Feature	Feature generated	Description
1	created_at	Account Age	time difference between account creation date and current date
2	tweet_url (number of tweets) and account_age	Posting Density	number of tweets posted divided by account age.
3	created_at	Activity Hour	determines the hours when users are most active.
4	retweet_count,	avg_retweet	average retweets per user
5	location	Unique Locations	number of unique locations listed by users
6	image_url	Image Count	number of images uploaded in tweets
7	favorite_count	avg_favorite	average number of favorites received per tweet.
8	reply_count,	avg_reply	average replies per user
9	quote_count	avg_quote	average quotes per user

$$A_{ge} = (T_{now} - T_{Created}).days \quad (1)$$

$$posting_{density}(\mu) = Count\ (tweets\ by\ \mu) \quad (2)$$

$$activity_{hour} = hour\ (T_{created}) \quad (3)$$

$$image_{count\ (\mu)} = count(d.image_{url} \neq null) \quad (4)$$

$$unique_{location\ (\mu)} = |set\ of\ location\ values\ for\ user\ \mu| \quad (5)$$

$$avg_{m(\mu)} = \frac{\sum_{d \in D, d.username=\mu} d.m}{posting_{density}} \quad (6)$$

$$avg_{favorite(\mu)} = \frac{\sum d.favorite\_count}{posting_{density(\mu)}} \quad (7)$$

by incorporating behavioral indicators such as posting density, favorite ratios, and interaction frequencies.

After labeling, data aggregation is performed carefully by combining feature data into feature vectors based on each account's activity, ensuring a comprehensive basis for analysis.

B. Feature Engineering

Feature engineering is the process of creating new features from raw data to improve the performance of machine learning models. It plays an essential role in helping models identify suspicious behavioral patterns in fake account detection on social media. This process involves analyzing data to determine features that provide deeper insights into user behavior, allowing differentiation between real and fake accounts [15].

One common approach in feature engineering is calculating metrics derived from user interactions with content. For example, features such as the number of retweets, comments, and likes on each tweet can indicate an account's engagement level. The ratio of total interactions (retweets, comments, and likes) to the number of followers can also help identify abnormal behavior, such as accounts showing high interaction levels despite having few followers.

Time extraction is another technique used in this process, particularly by analyzing the *created_at* feature. This step extracts specific temporal information from data containing date and time, which is then grouped and analyzed by intervals such as daily, weekly, or monthly to identify trends and behavioral patterns.

Metadata also plays a valuable role in feature engineering. It provides supplementary information about tweets, including posting time, posting frequency, and activity distribution over time. For example, fake accounts may exhibit unusually high activity during specific hours, which is atypical for genuine users. By integrating such features, machine learning models become more effective in detecting suspicious accounts [16].

Overall, effective feature engineering enhances the accuracy of fake account detection models and contributes to a better understanding of user behavior on social media. Therefore, continuous exploration

and innovation in feature engineering techniques are crucial for improving detection systems on platforms such as Twitter and Facebook, as well as preserving the integrity of information in cyberspace.

Feature engineering in this study focuses on analyzing user behavior. The processes performed at this stage include calculating account age, posting density, activity hour, image count, unique location, average interaction, and average favorite. The detailed description of feature engineering is presented in Table 2.

Account age, denoted as A_{ge}, is calculated by comparing the account creation date ($T_{created}$) with the current date T_{now}. The creation date is converted to timezone-naive using tz_localize (None). The difference between the current date and the account creation date is measured in days, indicating how long the user has been registered. The account age is calculated as shown in (1).

Posting density ($posting_{density}$) refers to the number of posts made by a user (μ). The calculation is performed based on the *username* column using the *groupby* method, and the total number of posts is counted using *size()*. This process generates a new column showing each user's total number of posts, which reflects their activity level. Let μ = username, then the calculation is expressed in (2).

Activity hour ($activity_{hour}$) represents the hours when users are most active, determined from the post creation time ($T_{created}$) by extracting the hour value from the *created_at* column using *dt.hour*. This process provides insights into user activity patterns. The calculation is shown in (3).

Image count, denoted as $image_{count(\mu)}$, is obtained by calculating the number of images uploaded by each user. The data are grouped by *username*, and the number of entries in the *image_url* column ($d.image\_url$) is counted. The calculation for $image_{count(\mu)}$ is shown in (4).

Unique location, denoted as $unique_{location(\mu)}$, is calculated by determining the number of distinct locations associated with each user. The process groups data by *username* and counts the unique location values. The formulation for $unique_{location(\mu)}$ is presented in (5).

Average interaction, denoted as $avg_{m(\mu)}$, represents the average number of replies, retweets, and quotes. To calculate the average reply (d), the data are grouped by *username* (denoted as μ), and the average number of replies is computed using the *mean()* method applied to the *reply_count* column. The resulting values are stored in a new column called *avg_reply*. The same procedure is used to calculate the average number of retweets and quotes using the *retweet_count* and *quote_count* columns, respectively. The final results are stored in new columns named *avg_reply*, *avg_retweet*, and *avg_quote*, which represent the user's average interaction, as shown in (6).

Average favorite, denoted as $avg_{favorite\ (\mu)}$, refers to the average number of likes received by a user's posts. This value is obtained by grouping data by *username* and calculating the average number of likes using the *mean()* method on the *favorite_count* column. The results are stored in a new column called *avg_favorite*, as shown in (7).

C. Data Split

The data are divided using the *train_test_split* function from the *scikit-learn.model_selection* library, which separates the dataset into two parts: the training set and the testing set. This process begins by defining the features (*X*) and the target (*y*). The *test_size* parameter determines the proportion of data allocated for testing; in this study, 30% of the data are used for testing and 70% for training. Additionally, the *random_state* parameter is set to ensure consistent data partitioning across runs. After splitting, *X_train* and *y_train* are used to train the model, while *X_test* and *y_test* are used to evaluate model performance after training.

D. Data Testing

The testing data were obtained using the *train_test_split* function from the *scikit-learn.model_selection* library. This process separates the resampled dataset, namely *X_resampled* for features and *y_resampled* for labels, into two parts: the training set and the testing set. Thirty percent of the data were allocated for testing, while the remaining seventy percent were used for training. The *random_state* parameter was set to 42.

$$\Gamma = \begin{Bmatrix} A_{ge}, posting_{density}(\mu), posting_{density}(\mu), activity_{hour}, \\ image_{count\ (\mu)}, unique_{location\ (\mu)}, avg_{m(\mu)}, avg_{favorite(\mu)} \end{Bmatrix} \quad (8)$$

$$\mathcal{M} = \{Random\ Forest, AdaBoost, SVM\} \quad (9)$$

$$h^{(t)}(x) = C_{L(x)} \quad (10)$$

$$h^{(t)}(x) = \sum_{L=1}^{N_{leaves}} C_L \,.\, 1\ (x \in R_L) \quad (11)$$

$$\hat{y} = MajorityVote\big(h^{(1)}(x), h^{(2)}(x), \dots, h^{(T)}(x),\big) \quad (12)$$

$$\hat{a} = sign\left(\sum_{t=1}^{T} \alpha^{(t)} \omega^{(t)}(x)\right) \quad (13)$$

$$\hat{s} = sign\ (w^T x + b) \quad (14)$$

$$\hat{s} = sign\left(\sum_{i=1}^{n} \alpha_i s_j\, K(x_i, x) + b\right) \quad (15)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (16)$$

$$Precision = \frac{TP}{TP + FP} \quad (17)$$

$$Recall = \frac{TP}{TP + FN} \quad (18)$$

E. Data Training

The training data were generated through dataset partitioning using the *train_test_split* function from the *scikit-learn.model_selection* library. In this study, the resampled dataset (*X_resampled* for features and *y_resampled* for labels) was divided into training and testing subsets. The *test_size* parameter was set to 0.3, meaning 30% of the total data were used for testing and 70% for training.

F. Training Model

In training the machine learning models, the Random Forest, Support Vector Machine (SVM), and AdaBoost algorithms were implemented using Google Colab with the Python programming language and the *scikit-learn 4.6.1* library. Random Forest is an ensemble algorithm that uses multiple decision trees to make predictions. According to Kaur and Singh [15], Random Forest is advantageous in handling large and complex datasets and helps reduce overfitting. In fake account detection, it is used to identify behavioral patterns that may not be evident to other algorithms.

AdaBoost is a boosting algorithm that combines several weak models to create a stronger one. It improves prediction accuracy by assigning greater weight to errors from previous iterations. Nguyen et al. [16] stated that using AdaBoost in fake account detection can enhance both the sensitivity and specificity of the model.

The hyperparameter settings were as follows: Random Forest (100 trees, unlimited depth), AdaBoost (50 estimators, learning rate = 1.0), and SVM (RBF kernel, C = 1.0, gamma = scale). Scalability was addressed through the use of parallel computation options available in Spark ML for larger datasets.

G. Classification

Classification is a technique in machine learning used to group data into specific categories or classes

based on their features. In this study, the classification models include SVM, Random Forest, and AdaBoost. The objective is to classify fake and real (genuine) accounts. The feature set is defined as follows in (8). A matrix $X \in \mathbb{R}^{N \times p}$ is constructed, where $p = |\Gamma|$. This matrix is used in the classification models denoted as $\mathcal{M}$, shown in (9). Random Forest and AdaBoost are extensions of the decision tree method, where each decision leaf prediction is calculated using (10). Here, $L(x)$ represents the index of the leaf node reached by splits based on i, and C_L is the predicted class or value stored in leaf L. This can also be expressed as shown in (11) where R_L denotes the region of the feature space belonging to leaf L, and $1(.)$ is the indicator function of the decision. The Random Forest formula is given in (12) where each $h^{(t)}$ represents one decision tree, and T is the total number of trees. The AdaBoost formula is defined in (13) where $\omega^{(t)}$ is the weak classifier, $\alpha^{(t)}$ is the weight of each weak classifier, and T is the total number of iterations. The Support Vector Machine (SVM) classifier is defined as follows in (14) where w is the vector of learned weights, b is the bias term, and x is the input feature vector. When using an RBF kernel, the formulation is shown in (15) where $K(15)$ represents the computed similarity between vectors.

H. Evaluation

At this stage, the model performance is evaluated based on accuracy, precision, recall, F1-score, the confusion matrix, and graphical visualization using the Precision–Recall Curve. In this study, true positives are categorized into two types: first, *Fake Identified Accounts*, which are accounts predicted by the model as fake and verified as fake based on existing data or labels; and second, *Accounts with Fake Characteristics*, referring to accounts that exhibit behaviors or features consistent with fake accounts, such as false identities or suspicious activity, and are correctly classified by the model.

It is essential to distinguish true positives from false positives, which are real accounts incorrectly identified as fake, and false negatives, which are fake accounts incorrectly identified as real. True positives indicate the classification model's effectiveness in accurately detecting fake accounts. Meanwhile, true negatives (TN) represent genuine accounts correctly identified as real by the model. In this study, true negatives consist of two types: *Genuine Identified Accounts*, which are accounts predicted and verified as genuine based on existing data or labels; and *Accounts Without Fake Characteristics*, referring to accounts that display normal user behavior, such as natural interactions, complete profile information, and no suspicious activity.

False positives indicate real account activities that are misclassified as fake, while false negatives represent fake account activities that are misclassified as real. Based on the confusion matrix, the model's performance is measured using accuracy, precision, and recall, as shown in (16)–(18).

III. Results and Discussion

The main objective of this study is to develop a detection model for identifying fake accounts on Twitter. Model optimization is performed through feature engineering, using data obtained from crawling and simulation-based collection. The crawling process through the Twitter API produced 1,219 tweets. The simulation process involved creating 100 fictitious accounts with varied characteristics. These accounts were generated by compiling a list of usernames (from @user001 to @user100), using royalty-free images as profile pictures, and writing diverse bios such as "Technology lover," "Environmental activist," and "Sports lover." Each account posted between 5 and 10 tweets related to its bio theme and used relevant, popular hashtags to attract other users.

The simulated accounts were also programmed to follow relevant users and interact with their content by liking and replying to tweets. This setup aimed to imitate the behavior of genuine users and establish a realistic fake social network. After these accounts engaged in interactions, data were collected by recording the type of content posted, the number of interactions, and responses from other users.

The data cleansing process reduced the dataset from 1,219 rows of tweet data containing 15 features to 812 rows with 14 features, representing a 33.9% reduction. The cleansing was carried out by removing the *conversation_id* column and duplicate entries, totaling 407 duplicate tweets. An example of the data cleansing results is shown in Figure 2.

In this study, the data transformation process focuses on modifying data types to ensure consistency.

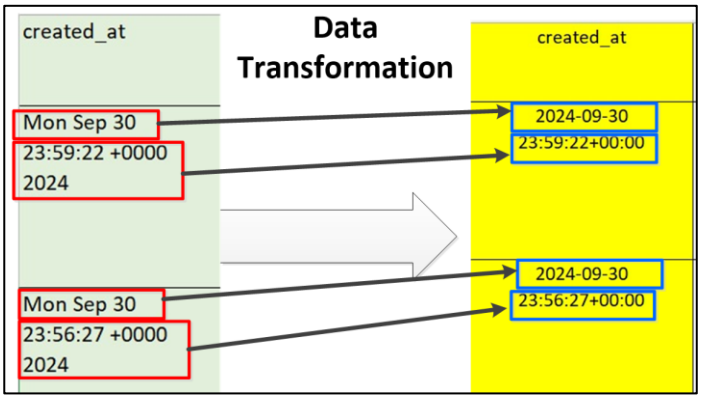


Figure 3 Results of data transformation changes in the created_at feature

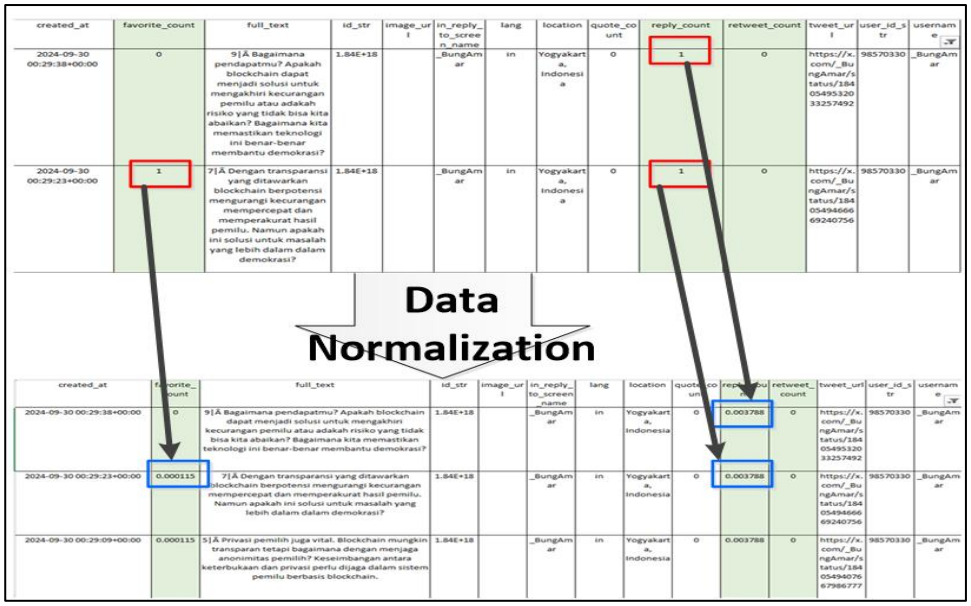


Figure 4. Results of Data Transformation

$$X' = \frac{X - X_{min}}{X_{Max} - X_{min}} \tag{19}$$

Specifically, the *created_at* column is converted from a string to a date data type, while the *favorite_count*, *reply_count*, and *retweet_count* features are changed from string to numeric formats. An example of the transformation applied to the *created_at* feature is shown in Figure 3.

The data normalization process standardizes the value of each feature using the MinMax Scaler approach, as shown in Equation (19), where X represents the original value, X_{min} is the minimum column value, X_{max} is the maximum column value, and X' denotes the normalized value. This normalization process scales each feature's value range between 0 and 1. The results of the normalization process are shown in Figure 4.

Labeling was performed manually for each tweet account in the dataset, involving expert assessment of 812 tweets. The subsequent data aggregation process combined expert-labeled data with simulated data identified as fake accounts. The results of the data merging process are presented in Table 3.

In this study, the machine learning classification models used include Random Forest, AdaBoost, and Support Vector Machine (SVM). During the data-splitting process, 70% of the data were used for training and 30% for testing. The data composition at the classification stage is summarized in Table 4.

The purpose of this study is to propose a feature engineering approach to improve the accuracy of detecting fake accounts on social media. Two evaluation approaches were conducted: detection testing without feature engineering and detection testing with feature engineering. In detection testing without feature engineering, the Random Forest model achieved the best performance, with an accuracy of

TABLE 3
MERGED DATA RESULTS

Data	Real account	Fake Account	Total Data
Twitter API Crawling	812	0	812
Simulation Data	0	100	100
Total Data	912		

TABLE 4
DATA SPLIT COMPOSITION

Total Data	Training Data (70%)	Testing Data (30%)
912	638	274

TABLE 5
COMPARISON OF CLASSIFICATION MODELS

Classification Model	Without Feature Engineering			With Feature Engineering (Proposed Model)		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Random Forest	98.77%	99.00%	99.00%	99.69%	99.00%	99.00%
AdaBoost	98.57%	99.00%	99.00%	99.94%	100.00%	100.00%
Support Vector Machine	95.90%	96.00%	96.00%	99.32%	99.00%	99.00%

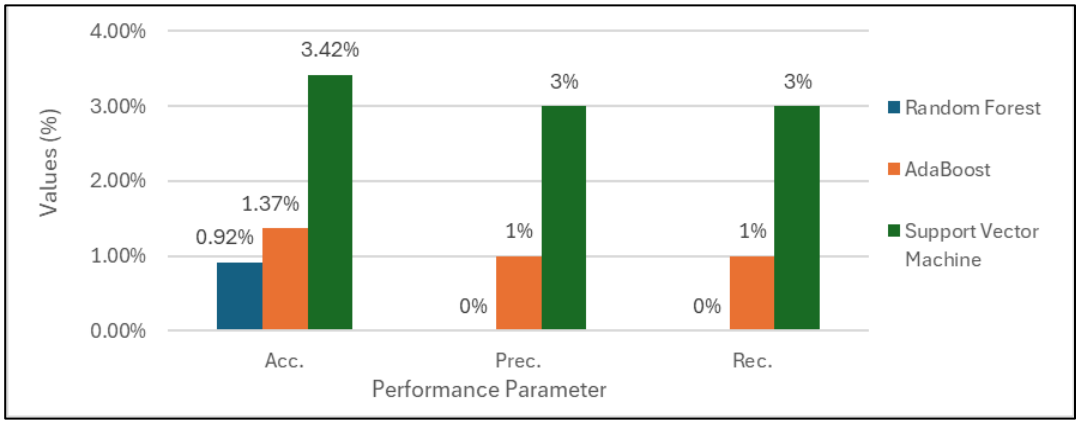


Figure. 5 Model Performance Improvement Analysis

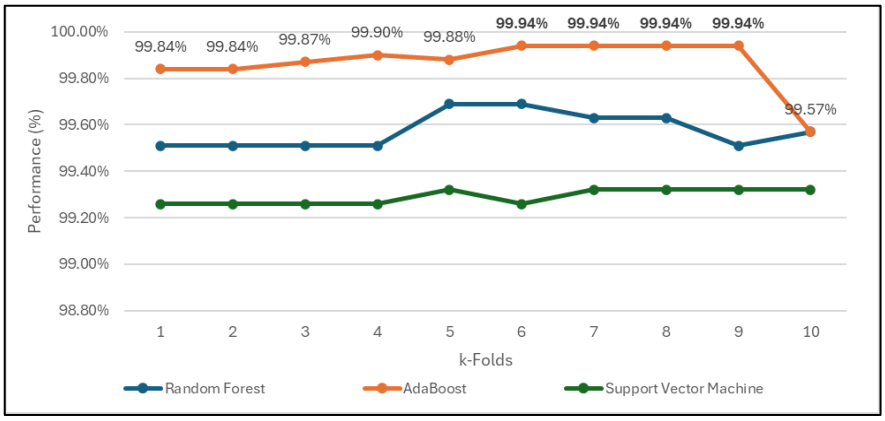
98.77%, precision of 99%, and recall of 99%. The lowest performance was observed in the AdaBoost model, which achieved an accuracy of 98.57% and precision and recall of 99%.

This paper proposes a feature engineering technique that generates new features from existing ones to enhance the performance of the detection model. The application of feature engineering results in an improvement across all classification models, with the highest increase observed in the SVM model. The SVM's accuracy improved from 95.90% to 99.32% (an increase of 3.42%), precision rose from 96% to 99% (an increase of 3%), and recall increased from 96% to 99% (an increase of 3%). The Random Forest model showed an accuracy improvement of 0.92%, while its precision and recall remained unchanged. The AdaBoost model demonstrated an accuracy increase of 1.37%, with both precision and recall improving by 1%. The performance improvements of each classification model are illustrated in Figure 5.

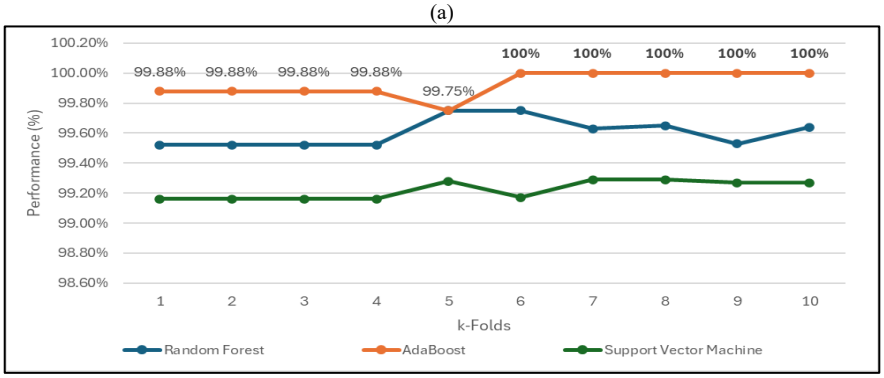
Among the three classification models, the AdaBoost model achieved the best detection performance. It reached an accuracy of 99.94%, precision of 100%, and recall of 100%. This superior performance became evident after implementing the feature engineering process. A detailed comparison of the classification models, both with and without feature engineering, is presented in Table 5.

In this study, the AdaBoost model's performance was further validated using K-fold cross-validation. This method, which tests the model with k values ranging from 1 to 10, provides a robust evaluation of its reliability and generalization capability. The test results, shown in Figure 6, demonstrate the model's consistency and stability across different k values. Figure 6(a) shows the accuracy. Figure 6(b) shows the precision. Figure 6(c) shows the recall.

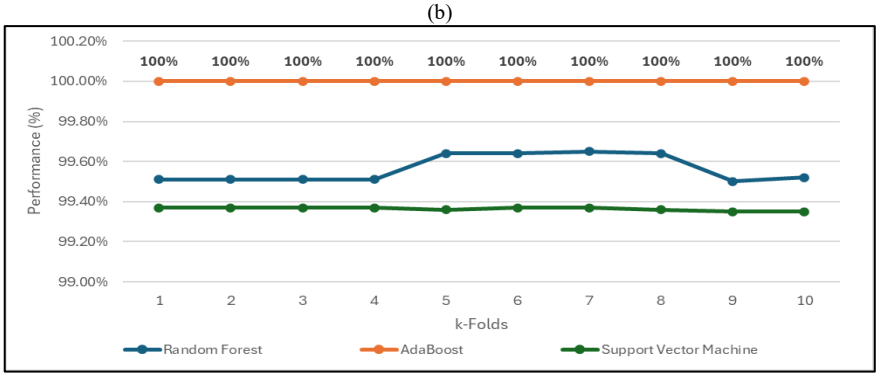
From the analysis of the k values, the improvement in the performance of the AdaBoost classification model, as indicated by the accuracy value, becomes noticeable at $k = 3$ and reaches its highest level at $k = 6$. However, when $k = 10$, the accuracy decreases significantly. This decline occurs because, at $k =$



(a)



(b)



(c)

Figure 6. K-fold validation analysis

10, the dataset division results in fewer fake account training samples, causing the AdaBoost model to fail to detect some of them. In terms of precision, the AdaBoost model records its lowest value at $k = 5$ and begins to improve optimally at $k = 6$, reaching a precision of 100%. The low value at $k = 5$ is due to several real account classes being misclassified as fake accounts. However, from $k = 6$ to $k = 10$, the model no longer produced detection errors for real accounts, and most importantly, no false positive values were found, ensuring high model accuracy.

Furthermore, the AdaBoost model successfully detected every real account across all k values, from $k = 1$ to $k = 10$. Compared with the other two models, Random Forest and SVM, the AdaBoost model achieved higher accuracy, precision, and recall across all k values in the k-fold validation. The supervised models demonstrated strong performance, with AdaBoost achieving an accuracy of 99.94%. Evaluation metrics included precision, recall, and F1-score, supported by confusion matrix analysis. PCA-based visualization confirmed a clear separation between fake and real accounts, while feature importance analysis identified account age, posting density, and interaction ratios as the most influential attributes. These findings demonstrate that supervised learning models enhanced through feature engineering outperform unsupervised clustering methods in practical scenarios for detecting fake accounts.

IV. CONCLUSION

This study proposes feature engineering techniques to enhance the performance of classification models, specifically Random Forest, AdaBoost, and Support Vector Machine (SVM). The feature engineering process involves generating new variables from basic features, including account age, posting density, activity hours, image count, unique location, average interaction, and average favorite. These techniques successfully improved the accuracy, precision, and recall values of each classification method compared with models that did not apply feature engineering.

The first experiment demonstrated that the Random Forest method produced the best classification model for detecting fake accounts, with an accuracy of 98.77%, precision of 98.57%, and recall of 95.90%. After applying feature engineering, the SVM model showed the highest improvement, with accuracy increasing from 95.90% to 99.32% (an increase of 3.42%), precision rising from 96% to 99% (an increase of 3%), and recall improving from 96% to 99% (an increase of 3%). The Random Forest and AdaBoost models experienced smaller performance gains, with Random Forest accuracy improving by 0.92% while precision and recall remained unchanged, and AdaBoost accuracy increasing by 1.37%, with precision and recall improving by 1%.

Using the feature engineering approach, the AdaBoost model achieved the best overall performance among the three methods, with an accuracy of 99.94%, precision of 100%, and recall of 100%. The optimal k value obtained from K-fold evaluation was $k = 6$. Overall, the model demonstrated excellent performance with feature engineering and can assist system analysts in detecting fake accounts, supporting efforts to address forensic cybercrime challenges, particularly in identifying fraudulent social media accounts.

For future research, behavioral graph-based analysis should be explored to analyze the distribution and correlation of fake account activities. This approach may help identify the perpetrators of misinformation on social media platforms such as Twitter (X). Such analysis could also be used to design an early warning detection model and support forensic investigations of social media accounts. This study demonstrates that feature-engineered supervised learning models significantly enhance fake account detection on Twitter (X). Although the results are promising, limitations include the relatively small dataset, dependence on simulated accounts, and the absence of network-level features. Future studies should focus on integrating graph-based anomaly detection, deep learning autoencoders, and adversarial bot generation techniques to improve model robustness and scalability.

REFERENCES

[1] M. Aljabri, R. Zagrouba, A. Shaahid, F. Alnasser, and D. M. Alomari, "Machine learning based social media bot detection: a comprehensive literature review," 2023.
[2] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using N-gram analysis and machine learning techniques," IEEE Intelligent Systems, vol. 34, no. 1, pp. 23-31, 2019.
[3] A. Kumar and A. Mehta, "Detection of Fake Accounts in Online Social Networks," Journal of Cybersecurity and Privacy, 2021.
[4] T. Nguyen, H. Tran, and D. Hoang, "Detecting Fake Accounts in Online Social Networks," IEEE Access, 2020.
[5] P. Azami and K. Passi, "Detecting Fake Accounts on Instagram Using Machine Learning and Hybrid Optimization Algorithms," *Algorithms*, vol. 17, no. 10, p. 425, 2024, doi: 10.3390/a17100425.
[6] A. Aboud, N. Rokbani, S. Mirjalili, A. Hussain, H. Chabchoub, and A. M. Alimi, "A Quantum Beta Distributed Multi-Objective Particle Swarm Optimization Algorithm for Twitter Fake Accounts Detection," 2022.
[7] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," Information Sciences, vol. 467, pp. 312-322, 2019.
[8] I. B. Irena and E. Setiawan, "Fake News (Hoax) Identification on Social Media Twitter using Decision Tree C4.5 Method," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 4, no. 4, pp. 711-716, 2020. doi:10.29207/resti.v4i4.2125.
[9] Y. M. Vianny and E. Setiawan, "Implementation of Rumor Detection on Twitter Using J48 Algorithm," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 4, no. 5, pp. 775-781, 2020. doi:10.29207/resti.v4i5.2059.
[10] A. R. I. Fauzy and E. B. Setiawan, "Detecting Fake News on Social Media Combined with the CNN Methods," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 7, no. 2, pp. 271-277, 2023. doi:10.29207/resti.v7i2.4889.
[11] M. Azabou, J. Park, and E. Ferrara, "Characterizing and detecting Twitter bots during the COVID-19 pandemic," in 2020 IEEE International Conference on Big Data (Big Data), pp. 2966-2973, 2020.
[12] J. Camacho-Collados and M. T. Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning," Journal of Artificial Intelligence Research, vol. 69, pp. 149-200, 2020.
[13] T. Tran and H. Nguyen, "Improving Fake Account Detection Using Hybrid Similarity Measures," Journal of Computer Science and Technology, 2022.
[14] E. Van Der Walt, "Using Machine Learning to Detect Fake Identities: Bots vs Humans," 2018.
[15] A. Kaur and D. Singh, "Random Forest: A Comprehensive Review of its Applications in Big Data Analytics," International Journal of Computer Applications, vol. 182, no. 20, pp. 1-7, 2019.
[16] H. T. Nguyen, T. L. Nguyen, and T. D. Nguyen, "Enhancing Fraud Detection using AdaBoost Algorithm," Journal of Information and

Telecommunication, vol. 4, no. 3, pp. 253-265, 2020.
[17] M. Al-Qurishi et al., "A survey on feature engineering for machine learning-based detection of online social network spam and fake accounts," Journal of Network and Computer Applications, vol. 174, p. 102890, 2021.
[18] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake Twitter followers," Decision Support Systems, vol. 80, pp. 56–71, 2015.
[19] E. Van der Walt and J. H. P. Eloff, "Protecting minors on social media platforms—A big data science experiment," in Proc. HPI Cloud Symp., 2015, pp. 1–78.
[20] Y. Wang and Y. Liu, "A comprehensive survey on hyperparameter optimization for machine learning," ACM Computing Surveys (CSUR), vol. 55, no. 6, pp. 1-36, 2022.
[21] G. Kurnia, "Choosing the Optimal Data Split for Machine Learning: 80/20 vs 70/30," Medium, 2024. [Online]. Available: https://medium.com/@gunkurnia/choosing-the-optimal-data-split-for-machine-learning-80-20-vs-70-30-0fd266710236. [Accessed: Apr. 20, 2025].
[22] Trivusi, "Data Splitting: Pengertian, Metode, dan Kegunaannya," 16 Sep. 2022. [Online]. Available: https://www.trivusi.web.id/2022/08/data-splitting.html. [Accessed: Apr. 20, 2025].
[23] A. Gholamy, V. Kreinovich, et al., "Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation," 2018. [Online]. Available: https://scholarworks.utep.edu/cs_techrep/1209/.
[24] A. B. Gupta, R. K. Gupta, and S. S. Mehta, "Machine Learning Techniques for Fake Account Detection in Social Networks," Journal of Network and Computer Applications, vol. 101, pp. 45-56, 2020.
[25] M. A. Z. Alzahrani, H. M. Alharbi, and A. H. Alzahrani, "A Survey on Machine Learning Techniques for Fake News Detection," IEEE Access, vol. 9, pp. 123456-123471, 2021.
[26] S. D. R. S. M. Ali, R. R. M. Alshahrani, and H. H. A. Alzahrani, "Bot Detection in Social Media: A Machine Learning Approach," in Proc. 2023 International Conference on Artificial Intelligence and Data Science (ICAIDS), pp. 1-6, 2023.
[27] J. Smith, R. Johnson, and L. Lee, "Enhancing Fake Account Detection Using Deep Learning Techniques," Journal of Cybersecurity and Privacy, vol. 5, no. 1, pp. 15-30, 2022.
[28] T. Chen, Y. Wang, and K. Liu, "A Comprehensive Review of Machine Learning Approaches for Fake News Detection," IEEE Transactions on Information Forensics and Security, vol. 17, pp. 123-135, 2022.