IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Enhancing Spam Comment Detection on Social Media with Emoji Feature and Post-Comment Pairs Approach using Ensemble Methods of Machine Learning

Antonius Rachmat Chrismanto[1], Anny Kartika Sari[2*], and Yohanes Suyanto[3]
[1,2,3] Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia
[1] Faculty of Information Technology, Universitas Kristen Duta Wacana, Indonesia

Corresponding author*: Anny Kartika Sari (email: a_kartikasari@ugm.ac.id).

ABSTRACT: Every time a well-known public figure posts something on social media, it encourages many users to comment. Unfortunately, not all comments are relevant to the post. Some are spam comments which can disrupt the overall flow of information. This research employed two strategies to address issues in text spam detection on social media. The first strategy was utilizing emojis that had been frequently discarded in many studies. In fact, many social media users use emojis to convey their intentions. The second strategy was utilizing stacked post-comment pairs, which was different from many spam detection systems that solely focused on comment-only data. The post-comment pairs were required to detect whether a comment was relevant (not spam) or spam based on the post context. This research used the SpamID-Pair dataset derived from social media for Indonesian spam comment detection. After a comprehensive investigation, the emoji-text feature, the stacked post-comment pairs, and ensemble voting could boost detection performance (in terms of accuracy and F1). Adding manual features also improved detection performance. Based on the experiment, the best stand-alone methods for spam comment detection are the SVM (RBF kernel) and the soft voting ensemble method for the best average performance.

INDEX TERMS spam detection, ensemble method, emoji feature, post-comment pair, social media.

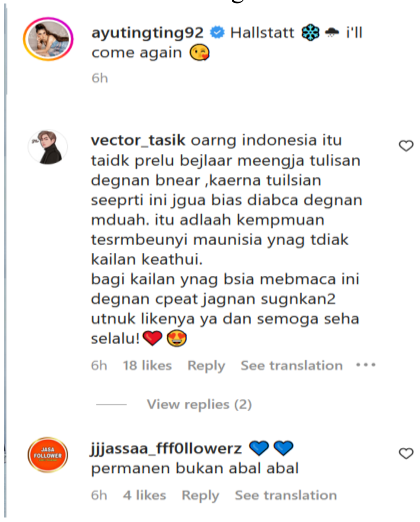
I. INTRODUCTION

Social media enables people to share their ideas and aspirations, collaborate, conduct business, promote products, and participate in politics. Popular social media platforms include Facebook (FB) for more formal or semi-formal text and image media, YouTube (YT) for semi-formal videos, Tik-Tok (TT) for non-formal videos, Instagram (IG) for semi-formal and non-formal text, images, and videos, and Twitter (TW) for semi-formal and non-formal text and images [1]. These social media have large user bases, are fully- and well-functioning, and are used by celebrities to increase their popularity.

Public figures who have large numbers of followers on social media include celebrities. Many celebrities utilize social media for promoting their activities, increasing their popularity, interacting with their followers, and other purposes. The more famous the celebrities are, the greater number of followers they have. With more followers, celebrities can interact with their fans more frequently [2]. As is characteristic of Web 2.0, users can now comment creatively on celebrities' feeds.

TW, YT, and IG are frequently used in spam detection research because these social media contain a lot of spam accounts and spam content. Particularly in Indonesia, spam content is usually found in comments against Indonesian artists, especially on IG [2]. Figure 1 depicts an example of a post and spam comments on social media in Indonesia of the @ayutingting account. Spam comments are very annoying and can disrupt the flow of information in the comments on a given post/status. Although some social media platforms already have spam filters, these are limited to English.

Another problem is the limited publicly available datasets for identifying spam text on social media. Most datasets on social media are found in English, and obtaining datasets in other languages, including Indonesian, is challenging. Many researchers conducted similar studies using their own collected datasets without sharing them.



ayutingting92 Hallstatt ❄️ ☔ i'll come again 😘
6h

vector_tasik oarng indonesia itu taidk prelu bejlaar meengja tulisan degnan bnear ,kaerna tuilsian seeprti ini jgua bias diabca degnan mduah. itu adlaah kempmuan tesrmbeunyi maunisia ynag tdiak kailan keathui.
bagi kailan ynag bsia mebmaca ini degnan cpeat jagnan sugnkan2 utnuk likenya ya dan semoga seha selalu!❤️😍
6h 18 likes Reply See translation ···

View replies (2)

jjjassaa_fff0llowerz 💙💙
permanen bukan abal abal
6h 4 likes Reply See translation

FIGURE 1. Example of A Public Figure's Post and Spam Comments on Social Media in Indonesia (https://www.instagram.com/p/CoRJyJgKaQP/)

SpamID-Pair[1] is a dataset provided for spam content detection in the Indonesian language available in Mendeley Data Repository. SpamID-Pair provides posts from Indonesian artists and their comments as pairs labeled spam/not spam. This dataset includes many emojis, which are widely used on social media. Users on social media frequently utilize emojis to describe their emotions and intentions. However, in various research in the Natural Language Processing (NLP) field, most emoji features are discarded/not used [3].

Studies of spam content detection have been previously conducted [4]–[9]. However, detecting spam content, particularly spam comments, is difficult due to multiple causes, for example: 1) the very unstructured and abnormal form of comment text; 2) the number of symbols and emoticons used by users; 3) the number of typos, intentional abbreviations, non-standard words, and mixed language usage; 4) some content is intentionally camouflaged to avoid being detected as spam, such as using the \/ sign instead of the letter V which becomes unreadable by the system; 5) the comments are spam but contain very subtle ads; and 6) the system fails to recognize the semantic meaning or semantic relationship between posts and comments. These issues are complex, require investigations, and necessitate many mutually supporting solution modules.

Some machine learning techniques in NLP can be used to identify spam comments. Based on [10], 14 best Machine Learning (ML) classification methods have been studied and compared, namely Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), Extreme Gradient Boosting (XGBoost), K-Nearest Neighbor (KNN), Ada Boost (AB), Naïve Bayes (NB), Multi-Layer Perceptron (MLP), and Decision Tree (DT). Machine learning techniques, also known as shallow learning techniques, are increasingly developing toward deep learning, which requires different learning techniques.

In this paper, the authors compared and explored the SpamID-Pair dataset collected from 12 celebrities with over 15 million followers [11] with different machine learning techniques according to [10] plus Complement Naïve Bayes (CNB) and Extra Tree (ET). This research made a contribution by providing comprehensive experimental results of spam detection performance (accuracy and F1) between non-emoji and emoji features with various combinations of hyperparameter scenarios (n-grams features, balanced/unbalanced data, the use of comment-only/post-comment pairs approach) using state-of-the-art machine learning and ensemble voting methods as well as their analysis [10]. This research also offers a new approach that uses post and comment text as pair-stacked input in machine learning to identify spam comments based on the posting context. This research uses NLP techniques on the Indonesian SpamID-Pair dataset.

The rest of the article is written as follows: 1) the introduction section that contains the background of spam on social media, the spam detection research problem, and our proposed research contribution; 2) the literature review section that includes up-to-date literature and theoretical references about spam detection using ML and ML algorithms; 3) the research methodology section that describes the scientific method used in this research, including the dataset used, pre-processing, implementation of 14 ML methods, and evaluation method; 4) the results-and-discussion section which describes the proposed ensemble models' experiments, results, analysis, and discussions; and 5) the conclusion section which explains our conclusion and suggestions for further research.

II. LITERATURE REVIEW

Some research on spam content detection has been conducted previously. Spam detection was mainly done in text messages [12], such as in the Short Message Services (SMS) [13], [14], which employed the UCI SMS dataset with the CNN method using auxiliary hand-engineered features [13]. Spam SMS was also detected using RNN-LSTM and LSTM only, which were also compared to machine learning methods [14]. Besides messages, there is much spam content on social media. Spam content can be found on social media like IG, FB, and TW [17].

Article [4] detected spam content based on spammers' accounts on IG in English. This study used Random Forest (RF) to detect the text content datasets totaling 1983 and 953808 media using their proposed method with special hand-engineered addition features. The significant hand-engineered features are a) the presence/absence of mention tags to another users; b) the hashtags number used,

[1] SPAMID-PAIR on Mendeley Data Repository (https://data.mendeley.com/datasets/fj5pbdf95t)

particularly the hashtags used that are not related to the content; c) the presence or absence of repeated words; d) specific keywords which tend to be spam as defined; and e) the presence/absence of watermarks on images. Using hand engineered features and k=10 in k-fold validation, the result reached 96.27%. Utilizing features that necessitated manual extraction was one of the limitations of the research.

The research [15] differed from [4] in that it employed Indonesian rather than English and did not detect spam posts but rather spam comments. The dataset used in [15] came from a publicly available dataset of Indonesian accounts. However, in contrast to what the authors did, the spam comments referenced in the study [15] were Indonesian-language comments with promotional purposes (such as advertising products). The combination of 1) keyword, 2) content text, and 3) hand-engineered features were employed. The handcrafted characteristics included the number of capital letters, the comment length, and the number of emoticons. Methods used in [15] did not use the emoji features. The keyword feature in the study consisted of specific keywords identified as selling/promoting particular products and extracted using an NLP regular expression pattern. Finally, the text features were extracted and weighted through various configurations of TFIDF, Bag of Words, and FastText techniques. Nave Bayes, SVM, and XGBoost were the classification methods used. Based on [15], it was found that using all of the features (features 1, 2, and 3) resulted in an F1 score of 96%. According to the research presented in [15], the employed characteristics were highly contingent on the dataset and cannot be applied to all new data, particularly for keywords retrieved using regular expressions.

Research on Indonesian spam comment detection, particularly on Instagram, was still rare. A study in [5] employed the Nave Bayes (NB) algorithm to detect Indonesian spam comments with a 72% accuracy rate. In contrast, [6] employed the opposite Nave Bayes algorithm, Complementary Naïve Bayes (CNB), because it used an unbalanced dataset between non spam and spam comments. With more non-spam comments than spam, the CNB algorithm could achieve an accuracy of 92%, while SVM only achieved 87%. Recent research on social media spam detection, including methods, results, datasets, emoji usage, and post context, is presented in Table I. Table I demonstrates that most researchers utilized privately compiled datasets.

SpamID-Pair is one of the available datasets and is taken from social media. The hallmark of this dataset is that it includes a large number of emojis that are included in the content. This dataset is also distinctive because the data consists of pairs of posts and comments labeled as spam or non-spam. The social media used in this dataset is IG. The reason is that IG is a popular social media with many users, and many public figures use it. Consequently, much spam is detected, especially in the comments of public figures on Instagram. IG data contains informal language, lots of emoticons/emojis, some of typos and abbreviations, lots of code mixes (mixed languages), comments of varying lengths but relatively short (1-3 sentences @ five words), a post-reply structure with no hierarchical data, and mention tags (using the symbol '@') [9].

TABLE I. RECENT RESEARCH OF SPAM DETECTION ON SOCIAL MEDIA

Methods	Language	Results	Datasets	Emoji and Post	Year
NB, SVM, XGB	INA	F1: 0.96 (SVM)	IG comments (private datasets) 24602 data	No	2017 [15]
RF	ENG	Acc: 0.96	IG profile (private dataset) 1983 profiles	No	2017 [4]
NB	INA	Acc: 0.77 (balanced)	IG comments (private dataset) 14500 data	No	2017 [16]
RF, SVM, NB	ENG	F1: 0.95 (SVM)	YT comments (private dataset) 13000	No	2018 [2]
AGA, ANN, SVM	ENG	Acc: 0.99 (AGA)	YT comments (private dataset)	No	2018 [17]
NB, LR	ENG	Acc: 0.87 (LR)	YT comments (private) 1956 data)	No	2019 [18]
NB, CNB	INA	F1:0.94 (CNB)	IG comments (private)	No	2019 [6]
RF, NB, DT	ENG	Acc: 0.90 (RF)	YT comment UCI	No	2019 [19]
LSTM, CNN	BGL	Acc: 0.95 (CNN)	Social Media	No	2019 [20]
NB	INA	F1: 0.83	IG comment (private) 700 data	No	2019 [5]
LR, DT, RF, AB, SVM	ENG	Acc: 0.95 (SVM)	YT comments (private) 400000 data	No	2020 [21]
KNN, DW-KNN	INA	Acc: 0.91 (DWKNN)	IG comments (private) 14500	No	2020 [8]
DT, KNN, SVC, GB, NB	ENG	Acc: 0.78 (NB)	FB comment (private) 2759 data, unbalanced	No	2021 [22]
CNN	INA	Acc: 0.97 (CNN multi modal)	IG posts image and text (private) 8000 data	No	2021 [23]

CART, LR, NB, RF, SVM, ANN, ESM	ENG	Acc: 0.95 (ESM)	YT comments (private) 6 million data	No	2021 [24]
DT, SVM, NB, RF, KNN	ARB	Acc: 0.84 (SVM)	YT comments (private) 40000 data	No	2022 [25]
SVM, RF	ENG	Acc: 0.95 (SVM)	YT comments on UCI 1956 data	No	2022 [26]
14 ML Methods (Ensemble Voting)	INA	Acc, F1	IG SpamID-Pair (public)	Yes	Our proposed (2023)

NB: Naïve Bayes; SVM: Support Vector Machine; XGB: eXtreme Gradient Boosting RF: Random Forest; AGA: Advanced Gradient; LR: Logistics Regression; CNB: Complement Naïve Bayes; DT: Decision Tree; LSTM: Long-sort Term Memory; AB: AdaBoost; KNN: K-Nearest Neighbor; DW-KNN: Distance Weighted KNN; GB: Gradient Boosting; CART: Decision Tree Variant; ANN: Artificial Neural Network; ESM: Ensemble Softmax.

The pre-processing phase was nearly identical to that of numerous studies that employed text data. NLP techniques were required for most pre-processing in detecting spam remarks or posts. Several references, such as [27]–[29], explained the importance of text pre-processing before further processing. Tokenization, case-folding, n-gram features, stemming, post-tagging, and stop-words removal were the methods that were used. Based on these pre-processing techniques, stemming techniques had the least significant effect. [29]. Besides pre-processing, most features in many NLP research features were the text. Some research used tokens feature in the form of BoW or weighted tokens in the form of TFIDF [30].

A. MACHINE LEARNING FOR TEXT CLASSIFICATION

There are two distinct approaches to machine learning: unsupervised and supervised learning. If it has problems with recognition or classification, it falls into supervised learning. However, this classification can also be developed using weakly-supervised or semi-supervised learning. The weakly supervised technique is based on the premise that unlabeled data can be labeled using only a small number of dataset labels and learning outcomes with a small number of labels. Several studies on weak supervision [22] and [23] also employed deep learning.

We primarily used machine learning methods from the best classification state-of-the-art methods from research [10]. We also combined a few other techniques, so there were 14 ML methods used in this research. These methods were the Multinomial NB method, Bernoulli Naïve Bayes (BNB), Complement Naïve Bayes (CNB), SVM Linear (SVML), SVM Radial Basis Function (SVMRBF), KNN (n=3), Decision Tree (DT), Random Forest (RF), Ada Boost (AB), XGBoost (XGB), Logistic Regression (LR), Extreme Tree (ET), Stochastic Gradient Descent (SGD), and Multi-Layer Perceptron (MLP). Detailed information about the techniques used in this study can be seen in Table IIIB.

Text spam detection belongs to text classification problems. As a text classification problem, we formulated a research problem as a document d as a document space (X) member, and there were fixed classes/labels $C = \{c_1, c_2, c_3 \ldots, c_n\}$. In spam detection/classification, the document space was typically high-dimensional. We were given a training set post-comment (PC) of a labeled document {d,c} where {d,c} was a member of X x C [31].

Naive Bayes is founded on Bayes' theorem and makes naive assumptions for each pair of features and class [32]. Theorem of Bayes where y is a class and x_1 through x_n can be formulated as (1):

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1,\ldots,x_n)} \qquad (1)$$

This formula assumes the naive conditions are independent as formula (2):

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y) \qquad (2)$$

NB predicts, for all data, whether x belongs to class y with the maximum posterior probability, according to the formula (3).

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,\ldots,x_n)} \qquad (3)$$

Since P(x_1,…,x_n) is constant, (3) can be simplified to formula (4) and formula (5) [33]:

$$P(y \mid x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y) \qquad (4)$$

$$\hat{y} = \underset{y}{arg(max)}\, P\ (y)\prod_{i=1}^{n} P(x_i \mid y))$$

Where

$$P(x_1, \ldots, x_n|y) = \frac{1}{\sigma_{ik}\sqrt{2\pi}}\, e^{\frac{(x_k - \mu_{ik})^2}{2\,\sigma_{ik}^2}}$$

is for continuous attributes.

The difference between Bernoulli Naïve Bayes (BNB) and Multinomial Naïve Bayes (MNB) is well suited for handling sorted text (documents), binary attributes, and multiple occurrences of tokens are ignored [31]. In addition, MNB is superior for handling larger texts, considering consecutive attributes and multiple occurrences of tokens. Compliment Naïve Bayes (CNB) is a multinomial NB variant suitable for working with non-uniform dataset distributions (imbalanced datasets). Instead of computing the probability that an item belongs to a particular class, CNB calculates the probability that an item belongs to all classes [34]. The CNB formula is derived from the formula MNB in formula (5), as seen in formula (6).

$$\hat{y} = \underset{y}{arg(max)}\, P\ (y)\prod_{i=1}^{n} \frac{1}{P(x_i|y)})$$

The SVM method is a technique that is considered to be very effective at classifying two classes (binary). It is memory efficient and has numerous kernel techniques that can be utilized in various situations. [35]. Vapnik presented the SVM algorithm in 1992 as a classifier algorithm based on a supervised learning technique. The SVM method seeks and locates an x-1-dimensional hyperplane to classify or categorize training data with multiple x attributes (the vector has x dimensions). The distance (margin) between classes must be maximized to locate the hyperplane. Consequently, SVM can guarantee that future data are extremely generalizable [36].

Assume that it is known that training data has been labeled and contains multiple *x* attributes (or pairs), (x_i, y_i) with $i = 1, 2, 3\ldots, n$, where n is the number of training data. While xi represents the set of attributes in the *i* and y_i training data is the class of *i* training data. SVM will calculate the optimization problem using equation (7) [37]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \ \sum_{i=1}^{x} \xi_i)$$

With the provisions according to formula (8):

$$y\_i\ (w\^T\ \phi(x\_i\ ) + b \geq 1 - \ \xi\_i\ \ , dan\ \xi\_i > 0. \quad (8)$$

Kernel function in SVM [33] is a transformation to determine the support vector so, which is learned in SVM as formulated as $K(X_i, X_j) = \ \Phi(x_i).\Phi(x_j)$. Linear kernel is formulated as $K(X_i, X_j) = x_k^T.x$ and radial basis function (RBF) as $K(X_i, X_j) = \exp\left\{-\frac{\|x-x_k\|_2^2}{\sigma^2}\right\}$.

K-Nearest Neighbor (KNN) is a type of supervised learning in which fresh data is classified based on the majority of the k-nearest neighbor category. As the predicted value for a new data value, the KNN algorithm employs Neighborhood Classification. The use of KNN in text classification is illustrated in [38], with an average accuracy of 95%.

KNN calculates the minimum distance between the data to be evaluated and the k closest nodes in the training data, where k is the number of nearest neighbors. The KNN algorithm consists of the following steps: 1) determining k, 2) calculating similarity / distance between the new and existing data, 3) sorting the distance by a threshold called k, and 4) selecting the class with the greatest number of members that has the nearest distance. The distance formula is found in equation (9).

$$d = \sqrt{(x_2 - \ x_1)^2 + (y_2 - \ y_1)^2} \quad (9)$$

A gradient-boosting algorithm is used for regression and classification problems. The components of this algorithm are a weak function, a weak learner, and an adaptive model. The loss function is highly dependent on the training dataset; weak learners can make predictions, and the additive model minimizes the loss function by incorporating weak learners.

A Decision Tree (DT) is a well-known method for classifying data that can be applied to complex problems [39]. Iterative Dichotomiser 3 (ID3), C4.5, which abolished the limitation of categorical features in ID3 by dynamically defining a discrete attribute that partitions the continuous attribute value into a discrete set of intervals, and CART (Classification and Regression Trees) are examples of DT algorithms. CART is comparable to C4.5, with the exception that it supports numerical target variables (regression) and does not compute rule sets [33]. CART generates binary trees employing the characteristic and threshold that produce the greatest information gain at each node. Gini Impurity is the Gini index used by CART for its splitting criterion. Scikit-learn employs a CART-optimized algorithm, but categorical variables are not presently supported [40].

All the classification methods described above are usually unstable and can be trapped in overfitting conditions. There are some ensemble learning methods. The main idea of this classifier is to use majority voting based on some ensemble methods. Some ensemble methods are bagging, boosting, stacking, and random forest (random ensemble). Boosting technique works to boost the weakest classifier algorithm [33].

Ada Boost is a meta-algorithm that evaluates the classifier on the original dataset and then modifies it using the same dataset. However, the weight of the incorrectly classified data is recalculated in order for the subsequent classifier to classify with greater precision [41]. The eXtreme Gradient Boosting (XGB) algorithm also includes a boosting component [42]. This algorithm combines models with limited precision in order to create a model with increased precision. The decision tree developed by Tianqi Chen functions as the basis for XGBoost. Since XGBoost was created as a library, it is compatible with a variety of programming languages, including Java, C++, Python, R, and Julia. Using L1 and L2 regularization, XGBoost supports SGD (Stochastic Gradient Boosting), Regular Gradient Boosting, and Regularized Gradient Boosting [43].

Random forest (RF) is a variant of the bagging technique in the ensemble methods. RF uses decision tree combinations, so each tree depends on random values from independent samples with uniform distribution. RF selects random features to partition each node to achieve high precision [33]. Additionally, the Extra Tree algorithm is founded on decision trees and ensembles of random forests. Extra Trees Classifiers, such as arbitrary Forest, make arbitrary decisions and randomize particular subsets of data to reduce overfitting and overlearning [44] [45]. Changeable parameters include the number of trees, features, and minimum size per split [44].

The ensemble ML method combines all the ML methods as training methods. It will get the best classifier by using each classifier and training each model on a different dataset sample. The prediction is made as majority voting using hard voting or weighted threshold majority voting for soft voting [46]. The ensemble voting will get the best parameters and advantages from all the ML methods so that the final voted method is returned and chosen as the final classifier [24].

The ensemble method is added as the new method to get the best classifier compared to the other methods.

B. MACHINE LEARNING EVALUATION

Three primary classification system processes exist: learning, validation, and evaluation. As shown in Table II below, a confusion matrix can be used to evaluate the system's performance and accuracy in classifying the dataset's sentiment. The confusion matrix depicts the performance of a classification system in terms of true positives, true negatives, false positives, and false negatives in order to calculate precision, recall, accuracy, and F1 score. In addition to the confusion matrix, the Area Under Curve (AUC) and the Receiver Operating Curve (ROC) can be used to determine the classification accuracy based on the true positive rate and false positive rate [47].

TABLE II. CONFUSION MATRIX

		Predicted	
		Negative	**Positive**
Real	Negative	True Negative	False Negative
	Positive	False Positive	True Positive

From the confusion matrix in Table II, additional calculations can be done to get the level of accuracy (accuracy) and f-measure in formulas (10) and (11).

$$Accuracy = (TN + TP) / (TN + FP + FN + TP) \quad (10)$$

$$F1\ Score = 2 * TP / (TP + FP + FN) \quad (11)$$

III. RESEARCH METHODOLOGY

The methodology proposed and carried out in this research is as follows (see also Figure 2):

1. Using and processing the SpamID-Pair dataset
2. Data exploration (profiling)
3. Pre-processing and data cleaning
4. Removing stop words
5. Normalization process
6. Implementing the spam comment detection algorithms according to Table IVA.
7. Experiment and evaluation based on the scenario in Table IVB.
8. Analysis, discussion, and conclusion stages.

Our research methodology is explained in more detail in the following sections.

A. SPAMID-PAIR DATASET

In this experiment, we used the SpamID-Pair dataset [48]. This dataset consisted of pairs of posts and comments from social media in Indonesian. The dataset contained 72874 data with spam or non-spam labels. Details of information on this dataset can be seen in Table III.

The characteristics of the SpamID-Pair dataset were: it consisted of repeated letters and symbols, included Unicode symbols, included emojis, contained non-standard/different abbreviations, had a lot of misspelled words, contained custom symbols, and contained code-mixing languages (Indonesian mixed with other languages).

TABLE III. SPAMID-PAIR DATASET PROFILE

IGID	Number of followers (millions)	Non-spam	Spam
1918078581	54.3	4565	2251
522969993	47.4	5712	1108
225064794	42.4	3397	691
24239929	36.4	818	1065
2993265	34.1	4528	2022
361869464	33.6	4658	1945
26444210	33.4	6854	2466
1948416	30.7	4944	1804
8115577	27.1	65	38
5735890	25.8	5045	1557
4934196	25.2	4818	1971
30585021	15.7	5537	911
		2896	1208

Data contains emoji/not.	Total	Percentage
Only text	22710	31,16
Contains Emoji	50164	68,84

Data is spam/not	Total	Percentage
Non-spam	53837	73.88
Spam	19037	26.12

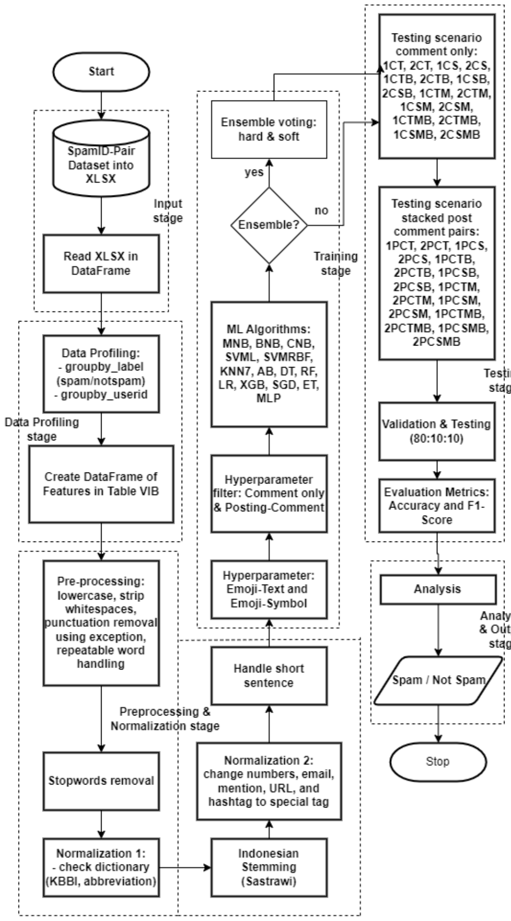

Start
SpamID-Pair Dataset into XLSX
Input stage
Read XLSX in DataFrame
Data Profiling: - groupby_label (spam/notspam) - groupby_userid
Data Profiling stage
Create DataFrame of Features in Table VIB
Pre-processing: lowercase, strip whitespaces, punctuation removal using exception, repeatable word handling
Preprocessing & Normalization stage
Stopwords removal
Normalization 1: - check dictionary (KBBI, abbreviation)
Indonesian Stemming (Sastrawi)
Normalization 2: change numbers, email, mention, URL, and hashtag to special tag
Handle short sentence
Hyperparameter: Emoji-Text and Emoji-Symbol
Hyperparameter filter: Comment only & Posting-Comment
ML Algorithms: MNB, BNB, CNB, SVML, SVMRBF, KNN7, AB, DT, RF, LR, XGB, SGD, ET, MLP
Ensemble?
yes
no
Ensemble voting: hard & soft
Training stage
Testing scenario comment only: 1CT, 2CT, 1CS, 2CS, 1CTB, 2CTB, 1CSB, 2CSB, 1CTM, 2CTM, 1CSM, 2CSM, 1CTMB, 2CTMB, 1CSMB, 2CSMB
Testing scenario stacked post comment pairs: 1PCT, 2PCT, 1PCS, 2PCS, 1PCTB, 2PCTB, 1PCSB, 2PCSB, 1PCTM, 2PCTM, 1PCSM, 2PCSM, 1PCTMB, 2PCTMB, 1PCSMB, 2PCSMB
Testing stage
Validation & Testing (80:10:10)
Evaluation Metrics: Accuracy and F1-Score
Analysis
Analysis & Output stage
Spam / Not Spam
Stop

FIGURE 2. Flowchart Of The Research Methodology

B. DATA EXPLORATION AND PRE-PROCESSING

Initial processing was carried out at this stage to explore, clean, and prepare the dataset for classification. Some pre-processing steps were:

a) Removing rows with NA/null.
b) Case folding: This process converted all the alphanumeric characters into lowercase characters.
c) Tokenization: This process split all sentences into words by using delimiter whitespaces. This tokenization scenario was carried out in 2 forms, 1-gram and 2-gram.
d) Text normalization: Text normalization converted all the tokens into "normal" tokens. The sklearn library handled this process. The SpamID-Pair dataset already provided data that was already normalized and also in raw format.
e) Stopwords elimination: This process eliminated all the stopwords from the Indonesian stopword list.

In this pre-processing step, we used Python libraries, such as Pandas and OpenPyXl for dataset manipulation, Matplotlib, and Seaborn for graphic and chart visualization, Tqdm for progress bar, and Sklearn as well as NLTK for text manipulation.

C. IMPLEMENTATION OF ML ALGORITHMS AND EVALUATION METRICS

Table IIIA shows the hardware and software utilized in this research. Due to limited resources, we made use of online machines in the cloud provided by AWS and offline on-premise machines. In accordance with [10] and two ensemble voting methods (soft and hard), various machine learning classification techniques were applied to process spam detection in this stage. Hard and soft ensemble methods took advantage of 14 ML methods and used the majority voting for the hard and weighted voting for the soft voting. All of the machine learning algorithms we used can be seen in Table IIIB. Table IIIB also displays the hyper-parameters (changed from the default or addition parameters) of the Scikit-learn library. The evaluations used in the case of spam comment detection were accuracy and F1-score. The reason we used F1-score was that the SpamID-Pair dataset was unbalanced, so using only accuracy was insufficient.

We used some Python libraries in this step, such as Scikit-learn, Pickle, and Matplotlib. Scikit-learn was employed to create TFIDF features in 1-gram, and 2-gram tokens, split the dataset into testing and training, implement the ML methods, and evaluate the classification result performance metrics. We used Pickle to save the trained model and load it again for testing.

We made use of four computers for the experiment, two were in the AWS cloud using SageMaker Studio Lab, and two were local computers using a Core i5 processor, 16 GB RAM, and 6 GB Nvidia RTX GPU. All code was generated in Jupyter Notebook. The TF-IDF feature was built from the SpamID-Pair text dataset with a maximum of 15000 features. All models were also saved so they could be reused for other implementations. Training duration varied from seconds, hours, to one day for each training method.

TABLE IVA. DEVICES SPECIFICATION AND FEATURES USED FOR THE EXPERIMENT

Information	Value
Hardware on-premise	
Processor	Core i5
RAM	16 GB
GPU	Nvidia RTX 6 GB
Standard cloud tool (Amazone SageMaker Studio Lab) https://studiolab.sagemaker.aws	
Features	TF-IDF weighted vector with max feature=15000, sub_linear=True
N-gram	1,2 grams
Balanced	Sklearn.SMOTETomek
Pre-processing	Tokenization, stopwords, normalization, stemming
Dataset	80% (70% (training) +10% (validation)) dan 20% (testing)
K-Fold:	10
Evaluation matrix	Confusion matrix (accuracy and F1 score)

TABLE IVB. TESTING PARAMETERS OF ML ALGORITHMS USED IN THE EXPERIMENT

ML Method	Parameter	Value
Naïve Bayes Multinomial (NB)	alpha	1.1
	alpha	1.1
Bernoulli Naïve Bayes (BNB)	binarize	0.51
Complement NB	alpha	1.1
	random_state	42
	dual	False
	penalty	l2
SVM Linear (SVML)	tol	0,0001
	kernel	RBF
	probability	True
	c	1.0
	gamma	Scale
SVM RBF (SVMRBF)	probability	True
	n_neighbors	7
	weights	distance
KNN7 (k=7)	metrics	euclidean
	n_estimators	1000
AdaBoost (AB)	random_state	42
	criterion	Entropy
	min_samples_split	3
Decision Tree (DT)	class_weight	{0:0.7}
	random_state	42
	warm_start	true
Random Forest (RF)	class_weight	{0,0.7}
	multi_class	ovr
	solver	saga
	max_iter	1000
Logistics Regression (LR)	random_state	42
	objective	binary-logistic
Xtreme Gradient Boosting (XGB)	random_state	42
	max_iter	1000
	tol	0.0001
Stochastics Gradient Descent (SGD)	alpha	0.0001
	verbose	0
	n_estimators	200
	random_state	42
	criterion	entropy
	min_samples_split	3
Extra Tree (ET)	class_weight	{0:0.7}
	random_state	42
	max_iter	300
Multilayer Perceptron (MLP)	verbose	False
Ensemble Voting from 14	voting	hard and

SotA methods (EV-H, EV-S)	soft

IV. RESULTS AND DISCUSSION

Based on the methodology described in the previous section, this study involved experiments on nine main topics, namely the effect of comment-only data without the emoji feature, the effect of post-comment pairs without the emoji feature, the effect of using emojis on comment-only data, the effect of using emojis on post-comment pairs, a comparison of performance against the usage of emojis on comment-only data and post-comment pairs, comparison of the performance of using emoji-text and emoji-symbols on comment-only data and post-comment pairs. The last part compared the stacked pair post-comment approach and the concatenated post-comment approach, manual features, and balanced scenario effect. The detailed discussion is presented below.

A. DATA NORMALIZATION, EMOJI HANDLING, AND THE USE OF MANUAL FEATURES

The normalization process was carried out after tokenization, as written in section III.B. The program was written in Python Jupyter Notebook and executed against the SpamID-Pair dataset. The *Kamus Besar Bahasa Indonesia* (the official dictionary of the Indonesian language) data consisted of 71798 word-class data (verb verbs, nouns, and adjective adjectives). In contrast, the dictionary data for abbreviations/acronyms/slang words was 1791 word pairs. The normalization process changed tokens that did not match the standard Indonesian spelling. The normalization method performed the following steps:

1. All tokens were matched with words in the dictionary. If it was not found in the dictionary, then the matching process was carried out with the abbreviation and slang word dictionary. If it was located in the dictionary of abbreviations, acronyms, and slang terms, the token was replaced with the appropriate token based on the dictionary.
2. All other tokens that were not found anywhere were left unchanged.
3. We removed punctuation in a list of "!$%&\+-<=>[\\]`{|}~" because it is related to emoji expressions.
4. We removed double letters in words such as “sayaaaa!!”, “cobaa…”, etc.).
5. We also converted some parts into special tagging with an UPPERCASE letter, such as URL pattern into HTTPURL tag, email pattern into EMAIL tag, user mentions into @USER tag, number pattern into ANGKA, and hashtag pattern into #HASHTAG tag.

For the emoji handling, we sent the processed tokens to the Demoji Python library and used the *demojize()* function that listed all converted emoji symbols to emoji text descriptions in plain English as the state in the standard UTF emoji table. We also made the scenario for the data without emojis with the Demoji library and removed all emojis returned by the *get_emoji_regexp()* function. Some examples of normalization and emoji text conversion can be seen in Table V.

TABLE V. NORMALIZATION AND EMOJI TEXT CONVERSION EXAMPLES

Original Text	Converted Text
KELUARIN SEMUA AGNEZ😢😍 POST!!!!	keluarin semua agnez crying_face smiling_face_with_heart-eyes post
Slmt siqng bini gw,yuk mkn siang,aku suapin pake rendang mauu??	selamat siqng bini gua yuk makan siang aku suapin pakai rendang mau
😍😍😍😍😍😍😍👏👏👏👏👏	smiling_face_with_heart-eyes smiling_face_with_heart-eyes smiling_face_with_heart-eyes clapping_hands clapping_hands clapping_hands
Wooowww😍🔥 . . Seediaf0llowerss guyss 🔥🔥	wooowww smiling_face_with_heart-eyes fire seediaf0llowerss guys fire fire

TF-IDF features are generated as follows: if the scenario is the comment only, we create TFIDF using the TfidfVectorizer from comment data and set max_features to 15000. If the scenario is post-comment, we create TFIDF from the post, TFIDF from the comment, and then stack horizontally. After that, we split TFIDF vector results into train and test data. These created vectors were *X_train* and *y_train, X_val and y_val, X_test* and *y_test.*

For the manual features, we used the lengths of the comments, lengths of both posts and comments, numbers of emojis in both posts and comments, numbers of unique emojis in both posts and comments, numbers of occurrences in both posts and comments, numbers of mention tags in both posts and comments, numbers of the hashtags in both posts and comments, numbers of capital letters in both posts and comments, numbers of link formats in both posts and comments, and, lastly, numbers of special characters in both posts and comments. To merge with the TF-IDF feature, we used *scipy.sparse* vector csr_matrix and created the horizontal stack of TFIDF features and all the additional manual features. We also applied a min_max scaling to these manual features before passing it to the classification method. We used the algorithm in data normalization, emoji handle, TFIDF generation, manual features, and the scenarios described in Algorithm 1.

We implemented 14 state-of-the-art models for the ensemble methods as the input with all the parameters in Table IVB. After the models were created and initialized, the VotingClassifier was also initialized with parameters, hard and soft. The voting classifier used majority voting models in the decision phase. The voting model was the biggest among the other models. After the voting model was created, it continued to the training-and-predicting stage. The algorithm can be seen in Algorithm 2.

Algorithm 1 Generate Features Method (TF-IDF, Emoji, balanced/non, and additional features)

```
Require: Dataset in XLSX format
Ensure: TF-IDF vectors
 1: Procedure GENERATEFEATURES(dataset)
 2:     df ← read_excel pandas(data)
 3:     df["comment"].replace('', NAN, inplace ← True)
 4:     kategori ← df["label"]
 5:     result ← pre_processing(df["comment"])
 6:     teks ← result
 7:     hasil ← list()
 8:     for word in teks.split() do
 9:         is-emoji ← bool(emoji.get emoji regexp().search(word))
10:         if is_emoji == False And is_ascii(word) then
11:             ketemu, pos1 ← cekKamus(kamus, word)
12:             if ketemu == False then
13:                 h ← correction(word)
14:                 word ← h
15:             end if
16:             word ← cekKamusSingkatan(kamus,singkatan,word)
17:             word ← re.sub('+','ANGKA',word)
18:             if word.islower() then
19:                 output ← stemmer.stem(word)
20:             else
21:                 output ← word
22:             end if
23:             if output not in stopwords then
24:                 hasil.append(output)
25:             end if
26:         else
27:             hasil.append(word)
28:         end if
29:     end for
30:     baru ← ' '.join(hasil)
31:     hasil_akhir ← emoji.demojize(str(baru),delimiters=(' ',' '))
32:     hasil_akhir ← ' '.join(hasil akhir.split())
33:     X ← hasil_akhir
34:     y ← kategori
35:     X_train, X_test,y_train,y_test ← train_test_split(X, y, test-size
    ← 0.20, random-state ← 42)
36:     Train_Y ← y_train; Test_Y ← y_test
37:     P ← X-train
38:     P['add_features_train'] ← X_train['add_feaatures']
39:     koloms1 ← ['add_features_train']
40:     P ← min_max_scaling(P, koloms1)
41:     add_features1 ← P['add_features_train']
42:     Train_X_transformed ← add_feature(Train_X_Tfidf, [add
    features1])
43:     P ← X_test
44:     P['add_features'] ← X_train['add_features']
45:     koloms2 ← ['add_features_test']
46:     P ← min_max_scaling(P, koloms2)
47:     kf ← KFold(n_splits ← 10, shuffle ← True,random_state ← 42)
48:     scorings ← ['accuracy', 'f1']
49:     Train_X_bal, Train_y_bal ← smotetomek.fit resample(Train
    X_transformed, Train_Y)
50:     Test_X_bal, Test_y_bal ← smotetomek.fit_resample(Test_X
    transformed, Test-Y)
51:     Train_X_Features ← [Train_X_bal or
    Train_X_transformed]
53:     Test_X_Features ← [Test_X bal or Test_X_transformed]
54:     Return: Train_X_Features, Test_X_Features, Train_Y,
    Test-Y
55: End Procedure
```

Algorithm 2 Ensemble Method Training and Testing)

```
Require: 14-ML models
Ensure: Hard and Soft Voting
 1: Procedure ENSEMBLELEARNING(MLModels)
 2:     list_of_models[] ← getModels(NBModel, BNBModel,
        CNBModel, SVMCModel, SVMRBFModel, KNN7Model,
        ABModel, DTModel, RFModel, LRModel, XGBModel,
        SGDModel, ETModel, MLPModel, VotingClassifier)
 3:     hard_voting ←
    VotingClassifier(estimator ← list_of_models,
    voting ← 'hard')
 4:     soft_voting ←
    VotingClassifier(estimator ← list_of_models,
    voting ← 'soft')
 5:     hard_model
    ←list_of_models['hard_voting']
 6:     hard_model.fit(Train-X-bal, Train-Y-bal)
 7:     soft_model ← list_of_models['soft_voting']
 8:     soft_model.fit(Train-X-bal, Train-Y-bal)
 9:     predictions-hard ← hard-model.predict(Test-X-bal)
10:     predictions-soft ← soft-model.predict(Test-X-bal)
11:     Return predictions-hard, prediction-soft
12: End procedure
```

B. THE EXPERIMENT RESULTS

The experiment results of spam comment detection using Machine Learning methods with various scenarios can be seen in Tables VIA and VIB. Table VIA shows that there were 14 ML methods used for testing spam comment data with multiple abbreviations. As shown in Table VIB, the scenarios were: using the TFIDF feature with 1-gram and 2-gram, comment-only data or posts and comment-combined data, non-emoji or emoji feature in Unicode symbols or text-converted emoji. Emoji conversion was done by changing the emoji symbols into the emoji descriptions according to the Unicode Table using the Demoji library. The emoji descriptions still used English text and a description separator in the form of an underscore character. In each result table, the highest values are written in bold, and the lowest ones are written in bold italics.

TABLE VIA. MACHINE LEARNING ABBREVIATION AND ITS DESCRIPTION USED IN THE EXPERIMENT

No.	Abbreviation Name	Description
1	NB	Multinomial Naïve Bayes
2	BNB	Bernoulli Naïve Bayes
3	CNB	Complement Naïve Bayes
4	SVML	SVM Linear
5	SVMRBF	SVM Radial Basis Function
6	KNN7	KNN with k = 3
7	AB	Ada Boost
8	DT	Decision Tree
9	RF	Random Forest
10	LR	Logistics Regression
11	XGB	eXtreme Gradient Boosting Tree
12	SGD	Stochastic Gradient Descent
13	ET	Extreme Tree
14	MLP	Multi-Layer Perceptron
15	EH	Ensemble Hard Voting
16	ES	Ensemble Soft Voting

TABLE VIB. TESTING SCENARIO ABBREVIATION AND MANUAL FEATURES

Scenario	Description	Scenario	Description
1CT	Features: token 1 gram, TFIDF, comment only, emoji text, pre-processing	1PCT	Features: token 1 gram, TFIDF, post-comment only, emoji text, pre-processing
2CT	Features: token 2 gram, TFIDF, comment only,	2PCT	Features: token 2 gram, TFIDF, post-

	emoji text, pre-processing		comment only, emoji text, pre-processing
1CS	Features: token 1 gram, TFIDF, comment-only emoji symbol, pre-processing	1PCS	Features: token 1 gram, TFIDF, post-comment only, emoji symbol, pre-processing
2CS	Features: token 2 gram, TFIDF, comment-only emoji symbol, pre-processing	2PCS	Features: token 2 gram, TFIDF, post-comment only, emoji symbol, pre-processing
1CTB	Features: token 1 gram, TFIDF, comment only, emoji text, pre-processing, balanced	1PCTB	Features: token 1 gram, TFIDF, post-comment, emoji text, pre-processing, balanced
2CTB	Features: token 2 gram, TFIDF, comment only, emoji text, pre-processing, balanced	2PCTB	Features: token 2 gram, TFIDF, post-comment, emoji text, pre-processing, balanced
1CSB	Features: token 1 gram, TFIDF, comment only, emoji symbol, pre-processing, balanced	1PCSB	Features: token 1 gram, TFIDF, post-comment, emoji symbol, pre-processing, balanced
2CSB	Features: token 2 gram, TFIDF, comment only, emoji symbol, pre-processing, balanced	2PCSB	Features: token 2 gram, TFIDF, post-comment, emoji symbol, pre-processing, balanced
1CTM	Features: token 1 gram, TFIDF, comment only, emoji text, pre-processing, add manual features	1PCTM	Features: token 1 gram, TFIDF, post-comment, emoji text, pre-processing, add manual features
2CTM	Feature 2 gram, comment text, emoji text, pre-processing, TFIDF, add manual features	2PCTM	Feature 2 gram, post-comment text, emoji text, pre-processing, TFIDF, add manual features
1CSM	Feature 1 gram, comment text, emoji symbol, pre-processing, TFIDF, add manual features	1PCSM	Feature 1 gram, post-comment text, emoji symbol, pre-processing, TFIDF, add manual features
2CSM	Feature 2 gram, comment text, emoji symbol, pre-processing, TFIDF, add manual features	2PCSM	Feature 2 gram, post-comment text, emoji symbol, pre-processing, TFIDF, add manual features
1CTMB	Feature 1 gram, comment text, emoji text, pre-processing, TFIDF, add manual features, balanced	1PCTMB	Feature 1 gram, post-comment, emoji text, pre-processing, TFIDF, add manual features, balanced
2CTMB	Feature 2 gram, comment text, emoji text, pre-processing, TFIDF, add manual features, balanced	2PCTMB	Feature 2 gram, post-comment text, emoji text, pre-processing, TFIDF, add manual features, balanced
1CSMB	Feature 1 gram, comment text, emoji symbol, pre-processing, TFIDF, add manual features, balanced	1PCSMB	Feature 1 gram, post-comment, emoji symbol, pre-processing, TFIDF, add manual features, balanced
2CSMB	Feature 2 gram, comment text, emoji symbol, pre-processing, TFIDF, add manual features, balanced	2PCSMB	Feature 2 gram, post-comment text, emoji symbol, pre-processing, TFIDF, add manual features, balanced
Manual Features:	length of the comment, length of both post and comment, number of emoji in both post and comment, number of unique emoji in both post and comment, number of number occurrences in both post and comment, number of mention tags in both post and comment, number of the hashtag in both post and comment, number of capital letters in both post and comment, number of link format in both post and comment, and the last, number of special characters in both post and comment		

1) SPAM DETECTION PERFORMANCE ON COMMENT DATA WITHOUT EMOJIS

Table VII displays the accuracy of the comment data only without using the emoji feature average (all the experiments use k-fold validation with k=10). The SVM-RBF kernel method produced the highest accuracy at 84%, while DT had the lowest accuracy at 63% in the 2CTMB scenario. The average accuracy across all scenarios was 78.46%. The CNB method was not executed when the scenario was a balanced dataset (which was generated using Sklearn.SMOTETomek library) because CNB is used in an unbalanced dataset. In all the tables, the cell is written as 'NA.' For example, it is written in Table VII for the 1CTB, 2CTB, 1CTMB, and 2CTMB scenarios. The best performance based on the scenario was 1CTB and 1CTMB using SVM-RBF, which achieved a score of 84%, followed by the SVM-Linear in the 1CTB scenario. Table VII also shows that SVMRBF seemed superior to the others, but Ensemble Soft Voting had the highest average accuracy of 82.375% compared to all other methods.

TABLE VII. THE AVERAGE ACCURACY OF COMMENT-ONLY DATA WITHOUT EMOJIS (IN PERCENT)

Accuracy	NB	BNB	CNB	SVML	RBF	KNN7	AB
1CT	79	73	79	79	82	73	74
2CT	79	72	78	78	81	74	74
1CTB	75	78	NA	82	**84**	74	72
2CTB	74	76	NA	82	83	74	72
1CTM	79	73	81	79	82	73	76
2CTM	79	72	80	79	81	74	76
1CTMB	80	78	NA	82	**84**	74	71
2CTMB	80	76	NA	82	**84**	74	71
AVG	78,13	74,75	79,50	80,38	82,63	73,75	73,25

Accuracy	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1CT	78	80	79	76	79	80	80	80	82
2CT	78	80	79	76	79	80	79	79	81
1CTB	81	81	82	77	82	82	82	83	83
2CTB	80	82	82	77	82	82	81	83	83
1CTM	73	79	79	78	80	80	80	82	82
2CTM	70	78	79	79	79	79	80	81	82
1CTMB	65	74	82	72	82	81	79	83	83
2CTMB	***63***	72	82	72	83	80	79	83	83
AVG	74	78	81	76	81	81	80	82	82

Table VIII displays the average F1 scores from the comment data without using the emoji feature. The SVM-

RBF method yielded the highest F1 score with the CTMB scenario. In contrast, DT earned the lowest F1 score. The average F1 score was 76.40%. The F1 score was also good because it was closer to accuracy. Based on the accuracy and F1 score, we can see that the best strategy for comment-only data was using the comment-text balanced and adding the manual features. The soft ensemble voting also had the highest average F1 score at 81% among all the other methods.

TABLE VIII. THE AVERAGE F1 SCORE OF COMMENT-ONLY DATA WITHOUT EMOJIS (IN PERCENT)

F1 Score	NB	BNB	CNB	SVML	RBF	KNN7	AB
1CT	73	64	75	75	79	72	67
2CT	73	63	75	74	79	72	67
1CTB	75	78	NA	82	**84**	74	70
2CTB	74	76	NA	82	83	74	71
1CTM	73	64	79	75	79	71	71
2CTM	74	63	78	75	79	73	72
1CTMB	80	78	NA	82	**84**	73	70
2CTMB	80	76	NA	82	**84**	74	70
Avg	75,25	70,25	76,75	78,38	81,38	72,88	69,75

F1 Score	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1CT	76	78	75	69	75	79	78	76	79
2CT	76	78	74	69	75	78	78	75	79
1CTB	81	81	82	76	82	82	82	83	83
2CTB	80	82	82	76	82	82	81	83	83
1CTM	71	76	75	74	76	78	78	78	79
2CTM	69	76	75	74	76	77	77	78	79
1CTMB	63	73	82	70	82	81	79	83	83
2CTMB	***59***	71	82	71	83	80	79	83	83
AVG	72	77	78	72	79	80	79	80	81

2) SPAM DETECTION PERFORMANCE ON POST-COMMENT PAIRS DATA WITHOUT EMOJIS

In this section, we evaluate the performance of spam comment detection using the post-comment pairs approach without emojis. All the emojis had been removed from this data. It contained only text data and was converted to TFIDF post-and-comment pairs stacked horizontally. Table IX displays the average accuracy of post-comment pair data without the emoji feature. The SVM-RBF methods produced the highest accuracy value at 86% using the SVM-RBF kernel in the 1PCTMB and 2PCTMB scenario, while DT got the lowest accuracy at 54% in 1PCTMB and 2PCTMB. The average accuracy value was 78.44%. The horizontally stacked TFIDF vectors of posts and comments differed only 0.02% from the average accuracy of comment-only data without emojis. Based on the ensemble methods, ES in post-comment pairs had higher accuracy than in comment-only data without emojis. ES ensemble also had the highest average accuracy among the other methods at 83.375%.

TABLE IX. THE AVERAGE ACCURACY OF POST-COMMENT PAIRS WITHOUT EMOJIS (IN PERCENT)

Accuracy	NB	BNB	CNB	SVML	RBF	KNN7	AB
1PCT	80	72	80	82	83	70	75
2PCT	80	72	79	81	83	68	74
1PCTB	78	76	NA	82	85	63	71
2PCTB	78	74	NA	82	85	62	72
1PCTM	79	75	80	82	83	72	77
2PCTM	79	73	80	82	83	69	77
1PCTMB	80	77	NA	83	**86**	64	71
2PCTMB	79	75	NA	83	**86**	61	71
AVG	79,13	74,25	79,75	82,13	84,25	66,13	73,50

Accuracy	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1PCT	74	75	82	77	81	78	80	82	82
2PCT	73	75	81	77	81	77	80	82	82
1PCTB	77	80	82	78	82	80	81	83	84
2PCTB	76	76	82	79	82	80	82	83	83
1PCTM	74	81	83	78	81	81	80	83	84
2PCTM	73	80	82	79	82	81	80	83	84
1PCTMB	***54***	80	83	73	83	83	82	84	84
2PCTMB	***54***	78	83	73	83	82	82	84	84
AVG	69	78	82	77	82	80	81	83	83

Table X shows the average F1 score from post-comment pairs data without emojis. The SVM-RBF method yielded the highest F1 score value. The average F1 score value reached 76.46%, an increase of +0.07% compared to the F1 score of comment-only data. The average value of the F1 score had the highest increment compared to its accuracy. This result indicates that post-comment can be horizontally stacked as pairs of data to improve spam detection performance. However, the average performance score of F1 Score without Emoji of post-and-comment pairs also indicates that it can and needs to be improved using the emoji feature and other scenarios. Based on the results of the study, it can be seen that the worst method was DT which reached the lowest value of 46%, followed by KNN and BNB. Ensemble ES got an F1 score which was higher than EH.

TABLE X. THE AVERAGE F1 SCORE OF POST– COMMENT PAIRS WITHOUT EMOJIS (IN PERCENT)

F1 Score	NB	BNB	CNB	SVML	RBF	KNN7	AB
1PCT	75	62	77	79	80	69	68
2PCT	75	62	77	79	80	68	68
1PCTB	78	76	NA	82	85	58	71
2PCTB	78	74	NA	82	85	57	72
1PCTM	74	70	78	80	80	70	72
2PCTM	75	63	78	80	80	68	72
1PCTMB	80	77	NA	83	**86**	60	70
2PCTMB	79	75	NA	83	**86**	57	70
AVG	76,75	69,88	77,50	81,00	82,75	63,38	70,38

F1 Score	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1PCT	73	74	79	71	78	76	77	79	80
2PCT	72	74	78	71	79	76	78	79	79
1PCTB	77	80	82	78	82	80	81	83	84
2PCTB	76	76	82	79	82	80	82	83	83
1PCTM	71	78	80	74	78	79	78	80	81
2PCTM	71	77	79	76	80	79	78	80	81
1PCTMB	***46***	80	83	72	83	83	82	84	84
2PCTMB	***46***	78	83	72	83	82	82	84	84
AVG	67	77	81	74	81	79	80	82	82

3) DETECTION PERFORMANCE ON COMMENT DATA WITH EMOJIS

In this section, we explore the detection performance on the comment-only data with emoji. We wanted to know how emojis can affect the performance of comment-only data. Based on the data in Table XI, it was found that the average accuracy of the comment-only data using the emoji feature was 79.82%. The SVM-RBF method yielded the highest accuracy values, which reached 88% (the highest until now) in 1CTMB scenarios. The DT method had the lowest accuracy at 51% in the 1CSMB scenario. It can also be seen that the emojis converted into the text format (emoji-text) had a higher value than the original emoji symbols in UTF-8 encoding (emoji-symbols). Interestingly, the performance of 1-gram and 2-gram token features with balanced data was the same as with non-balanced data. The ES method also performed better than EH in terms of accuracy, except in the CSMB scenario.

TABLE XI. THE AVERAGE ACCURACY OF COMMENT-ONLY DATA WITH EMOJIS (IN PERCENT)

Accuracy	NB	BNB	CNB	SVML	RBF	KNN7	AB
1CT	83	78	83	87	87	77	81
2CT	83	78	82	86	87	77	81
1CS	82	82	83	83	84	81	80
2CS	81	81	83	83	83	80	80
1CTB	79	80	NA	84	87	79	82
2CTB	78	76	NA	84	86	78	81
1CSB	73	65	NA	72	72	70	67
2CSB	73	64	NA	68	72	70	68
1CTM	83	78	85	87	87	77	83
2CTM	83	78	84	86	87	76	81
1CSM	81	79	85	83	84	77	79
2CSM	79	78	79	83	76	79	80
1CTMB	82	80	NA	86	**88**	79	77
2CTMB	82	76	NA	85	87	78	76
1CSMB	78	65	NA	72	76	77	60
2CSMB	78	64	NA	71	76	69	72
AVG	79,9	75,1	83,0	81,3	82,4	76,5	76,8

Accuracy	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1CT	83	86	86	83	86	86	86	87	87
2CT	83	86	86	83	86	86	85	87	87
1CS	81	83	83	81	83	83	82	84	84
2CS	80	82	83	81	83	82	82	83	83
1CTB	83	86	85	84	83	86	85	86	86
2CTB	83	85	85	83	85	85	85	86	86
1CSB	73	75	72	71	72	76	75	74	76
2CSB	73	74	72	71	72	75	74	74	75
1CTM	79	86	87	85	87	87	86	87	87
2CTM	78	86	86	84	86	86	85	87	87
1CSM	70	80	83	82	83	78	84	84	85
2CSM	79	84	78	84	76	84	84	84	84
1CTMB	67	79	86	71	86	87	83	86	86
2CTMB	67	75	86	72	86	82	83	86	85
1CSMB	***51***	57	75	***51***	75	67	79	81	67
2CSMB	55	64	75	56	74	66	79	81	72
AVG	74	79	82	76	81	81	82	84	82

Based on the information in Table XII, it was found that the average F1 score from comment-only data using the emoji feature was 75.33%. The SVM-RBF method also yielded the highest F1-score value. In the case of balanced emoji symbols, the DT methods had decreased performance significantly compared to text emojis until it reached 37%. Ensemble soft voting also performed the best on average compared to the other methods.

TABLE XII. THE AVERAGE F1 SCORE OF COMMENT-ONLY DATA WITH EMOJIS (IN PERCENT)

F1 Score	NB	BNB	CNB	SVML	RBF	KNN7	AB
1CT	74	65	79	82	82	74	70
2CT	75	64	77	81	81	74	70
1CS	69	69	76	74	75	68	69
2CS	68	68	75	74	74	67	68
1CTB	78	80	NA	84	87	79	82
2CTB	78	76	NA	84	86	78	81
1CSB	72	63	NA	71	70	70	65
2CSB	72	62	NA	66	70	70	65
1CTM	74	66	81	82	82	74	75
2CTM	75	65	80	82	81	73	72
1CSM	68	64	79	74	75	72	69
2CSM	70	70	73	74	53	71	69
1CTMB	82	80	NA	86	**88**	79	76
2CTMB	82	76	NA	85	87	77	75
1CSMB	77	63	NA	71	75	77	56
2CSMB	78	62	NA	70	75	68	72
AVG	74,5	68,3	77,5	77,5	77,6	73,2	70,9

F1 Score	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1CT	78	81	81	73	82	81	81	82	82
2CT	78	81	81	73	81	81	80	81	82
1CS	73	75	74	70	74	75	75	74	75
2CS	72	73	73	68	73	74	74	74	74
1CTB	83	86	85	84	83	86	85	86	86
2CTB	83	85	85	83	85	85	85	86	86
1CSB	72	75	71	70	70	75	74	73	75
2CSB	72	73	71	69	71	75	73	73	75
1CTM	75	81	82	79	82	82	81	83	83
2CTM	73	81	81	78	82	82	81	82	82
1CSM	65	75	73	75	73	73	78	75	77
2CSM	74	78	64	76	64	78	77	76	76
1CTMB	64	78	86	69	86	87	83	86	86
2CTMB	65	75	86	71	86	82	83	86	85
1CSMB	***37***	49	75	37	74	65	79	81	65
2CSMB	48	61	74	47	74	64	79	81	71
AVG	70	75	78	70	78	78	79	80	79

4) PERFORMANCE TESTING ON POST-COMMENT PAIRS DATA WITH EMOJIS

After experimenting with comment-only data with emojis, we continued testing the performance on post-and-comment pairs with emojis. Table XIII displays that the average accuracy of post-comment pairs data using the emoji feature was 80.36%. The SVM-RBF method with a 1PCTMB scenario yielded the highest accuracy value at 90% (the best accuracy so far). Still the same with comment-only data with emojis, emoji text produced a better result than emoji symbols in UTF-8 encoding. Based on these results, the accuracy of the stacked post-comment pairs data with emojis was higher than the comment-only data with emojis, reaching only 79.81%. It increased by 0.6%. This result was also better than the accuracy of post-comment pairs data with no emoji (only 78.42%), and the accuracy of comment-only data without emojis (78.49%). It increased by 1.94% and 1.87%. The DT method reached the worst accuracy with a 1CSMB scenario at 52%, and the ensemble ES was better than EH in the average accuracy at 84.875%. The ensemble methods could not outperform the single classifier but always yielded the highest result in average accuracy among the others.

TABLE XIII. THE ACCURACY OF POST-COMMENT PAIRS DATA WITH EMOJIS (IN PERCENT)

Accuracy	NB	BNB	CNB	SVML	RBF	KNN7	AB
1PCT	83	79	84	87	88	80	81
2PCT	83	83	83	87	87	78	82

1PCS	81	81	81	83	83	76	80
2PCS	82	78	80	83	83	76	80
1PCTB	81	80	NA	85	89	74	81
2PCTB	81	76	NA	85	89	72	81
1PCSB	73	68	NA	77	78	68	69
2PCSB	74	66	NA	77	78	67	69
1PCTM	83	78	84	87	88	80	83
2PCTM	83	78	83	87	87	79	82
1PCSM	82	78	81	84	84	77	77
2PCSM	82	78	80	83	84	77	77
1PCTMB	81	79	NA	86	**90**	75	76
2PCTMB	81	76	NA	85	89	72	76
1PCSMB	75	68	NA	77	78	67	57
2PCSMB	74	66	NA	77	78	68	75
AVG	79,9	75,8	82,0	83,1	84,6	74,1	76,6

Accuracy	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1CT	81	84	87	84	87	85	86	87	87
2CT	82	83	87	84	87	84	85	87	87
1CS	79	82	83	82	83	82	82	83	83
2CS	79	82	83	82	83	82	82	83	83
1CTB	81	84	85	86	85	85	85	87	88
2CTB	82	84	85	85	85	84	85	86	87
1CSB	76	80	77	77	77	80	80	79	80
2CSB	76	79	77	77	77	80	79	79	80
1CTM	81	85	87	85	87	86	84	87	88
2CTM	79	85	87	85	87	86	86	87	88
1CSM	73	83	84	83	84	83	83	84	85
2CSM	73	83	84	82	84	83	83	84	85
1CTMB	66	84	86	74	86	86	86	87	87
2CTMB	69	83	85	71	86	85	85	86	87
1CSMB	***52***	81	77	57	77	82	81	79	81
2CSMB	54	82	77	70	77	82	81	80	82
AVG	74	83	83	79	83	83	83	84	85

Table XIV shows that the average F1 score from post-comment data using the emoji feature was 75.86%. The SVM-RBF method still produced the highest F1-score value at 88% in all balanced emoji text scenarios. On the other hand, the DT method performed worst at just 52%. These results demonstrate an increase in F1-score compared to comment-only data with emojis but a very slight decrease in comment-only and post-comment pairs with emojis. This result means that the post-comment pairs approach and the emoji feature strongly influence the spam comment detection performance. We can see that the emoji feature had a higher impact than the post-comment pairs approach. Until this step, the converted emoji text was superior to the emoji symbols. As usual, the soft ensemble soft voting had the highest average F1 score among the other methods.

TABLE XIV. THE F1 SCORE OF POST-COMMENT PAIRS WITH EMOJI (IN PERCENT)

F1 Score	NB	BNB	CNB	SVML	RBF	KNN7	AB
1PCT	74	65	79	82	82	74	70
2PCT	75	64	77	81	81	74	70
1PCS	69	69	76	74	75	68	69
2PCS	68	68	75	74	74	67	68
1PCTB	78	80	NA	84	87	79	82
2PCTB	78	76	NA	84	86	78	81
1PCSB	72	63	NA	71	70	70	65
2PCSB	72	62	NA	66	70	70	65
1PCTM	74	66	81	82	82	74	75
2PCTM	75	65	80	82	81	73	72
1PCSM	68	64	79	74	75	72	69
2PCSM	70	70	73	74	53	71	69
1PCTMB	82	80	NA	86	**88**	79	76
2PCTMB	82	76	NA	85	87	77	75
1PCSMB	77	63	NA	71	75	77	56
2PCSMB	78	62	NA	70	75	68	72
AVG	74,5	68,3	77,5	77,5	77,6	73,2	70,9

F1 Score	DT	RF	LR	XGB	SGD	ET	MLP	EH	ES
1CT	78	81	81	73	82	81	81	82	82
2CT	78	81	81	73	81	81	80	81	82
1CS	73	75	74	70	74	75	75	74	75
2CS	72	73	73	68	73	74	74	74	74
1CTB	83	86	85	84	83	86	85	86	86
2CTB	83	85	85	83	85	85	85	86	86
1CSB	72	75	71	70	70	75	74	73	75
2CSB	72	73	71	69	71	75	73	73	75
1CTM	75	81	82	79	82	82	81	83	83
2CTM	73	81	81	78	82	82	81	82	82
1CSM	65	75	73	75	73	73	78	75	77
2CSM	74	78	64	76	64	78	77	76	76
1CTMB	64	78	86	69	86	87	83	86	86
2CTMB	65	75	86	71	86	82	83	86	85
1CSMB	**37**	49	75	37	74	65	79	81	65
2CSMB	48	61	74	47	74	64	79	81	71
AVG	70	75	78	70	78	78	79	80	79

5) PERFORMANCE COMPARISON ON COMMENT DATA WITH AND WITHOUT EMOJI SCENARIO

This section compares the detection performance between comment-only data with and without emojis. Figure 3 shows the increment of accuracy between comment-only data with and without emojis scenarios. Based on the results, it can be determined that the average increment in accuracy reached +5.97%, with the highest average improvement results obtained from the Ada Boost (AB) (+9.57%). RF followed it with +6.86%. AB achieved the most considerable average improvement in accuracy of +13.89%. In contrast, the XGB method obtained the lowest increment (decreasing to -1.39%). Ensemble hard voting had a higher increment than soft voting on average accuracy.

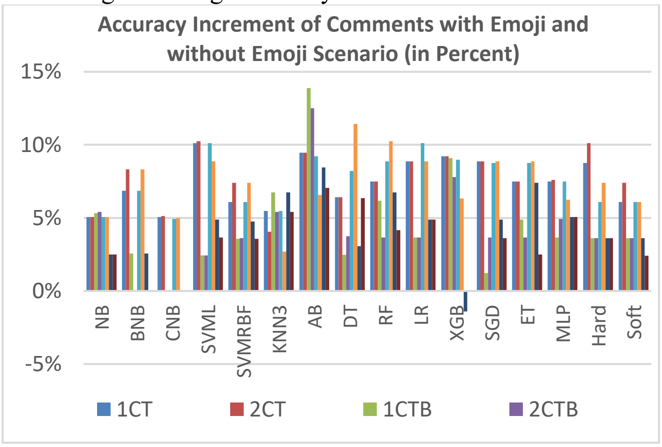


FIGURE 3. Accuracy Increment of Comments only with Emoji and without Emoji Scenario (in Percent)

On the other hand, figure 4 shows the increment of the F1 score between comments only with emojis and without emojis. Based on this figure, it can be seen that the average increment in the F1 score reached +4.68%. The highest average improvement results were obtained from the AB value at +7.69%. AB also received the best F1-score improvement with a +17.14% increment (1CTB). On the other hand, DT with a 1CTMB scenario got the worst increment with a decrement until -1.43%. The EH method got a higher F1 score than ES. The experiment result shows that the emoji features improved their average accuracy and F1-score in a range between +4.67% and +5.97%. Moreover, emoji usage improved spam comment detection performance, particularly in accuracy.

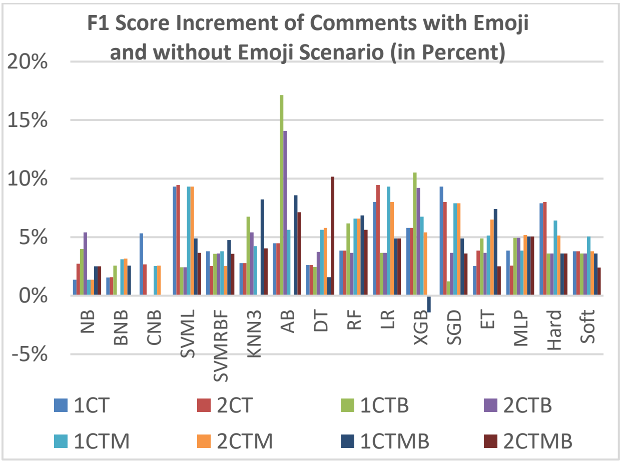


FIGURE 4. **F1-Score Comparison Between Comments with Emoji and Without Emoji (in Percent)**

6) PERFORMANCE COMPARISON ON POST-COMMENT PAIRS WITH AND WITHOUT EMOJI SCENARIO

This section compares the detection performance between post-comment pairs with and without emojis. Figure 5 shows the increment of accuracy between post-comment data without emojis and with emojis scenarios. Based on the result, it can be determined that the average increment in accuracy reached +6.64%. It was higher than the improvement of comment-only data in the previous result. Surprisingly, the highest average accuracy improvement results were obtained by DT with +27.78%, and the lowest average accuracy improvement was obtained by XGB (-2.74%). The highest improvement method was DT; meanwhile, the lowest was XGB, both with 2PCTMB scenarios. The emoji feature on post-comment pairs data improved spam detection accuracy. Ensemble soft voting performed better than hard voting in average accuracy increment.

Figure 6 shows the F1-score increment of post-comment pairs data with emojis and without emojis scenarios. Based on this result, it can be seen that the average increment in F1-score reached the value of +4.65%, with the most considerable improvement achieved by DT. The highest scenario was obtained by DT (on 2PCTMB), while BNB (on 1PCTM scenario) received the lowest F1 score. The average accuracy increment was higher than the average F1 score increment. The ES method had a higher F1 score increment than EH.

Figures 5 and 6 show that the accuracy and F1-score using the emoji feature in post-comment pairs data were higher than those without using the emoji feature. The increment of the average F1 score was between +4.65% and +6.64%, higher than the increment of the comment-only data. Stacked post-comment pairs improved the performance compared to just using comment-only data. So, it can be stated that emojis and post-comment pairs are excellent combinations for improving spam detection performance. The methods with the most significant improvement due to the emoji feature were DT and AB. XGB and AB typically had the lowest performance in the without-emoji-feature scenario, but using the emoji feature helped them improve their performance.

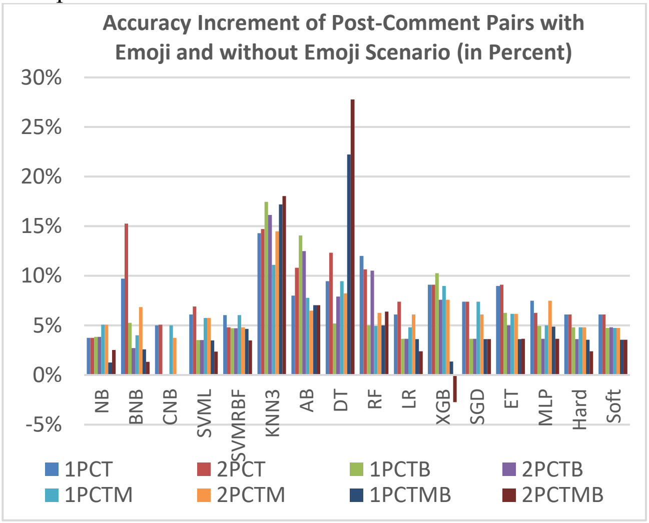


FIGURE 5. **Accuracy Increment of Post-Comment Pairs with Emoji and Without Emoji Scenario (in Percent)**

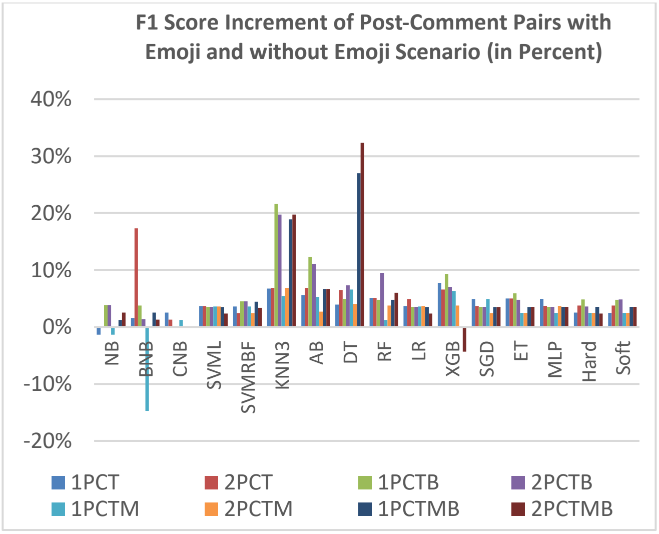


FIGURE 6. **F1-Score Increment of Post-Comment Pairs with Emoji and Without Emoji Scenario (in Percent)**

7) PERFORMANCE COMPARISON BETWEEN EMOJI POST COMMENT PAIRS AND EMOJI COMMENTS ONLY

Based on the previous section, the emoji feature improved spam detection performance. This section also shows the performance increment of emojis in comments and post-comment pairs scenarios. Based on the results in Figure 7, the average accuracy increment between emoji features in post-comments according to the methods was +1.53% and +1.67% according to the scenarios. The best methods that gained the most improvement were RF, ET, and ES. The KNN and DT experienced a decrease of -12.79% and -7.59%, respectively. KNN and DT based on the Tree algorithm could not perform well, even when using emoji features.

Interestingly, scenarios 1CSB, 2CSB, 1CSMB, and 2CSMB produced the best results compared to those of other scenarios. Emoji symbols were found to produce a higher increase in the result than emoji text when compared with comment-only data and post-comment pairs. The emoji symbols yielded promising results in accuracy when combined with post-comment pairs data. Ensemble with soft voting got a higher increment compared to hard voting.

The average F1-score comparison between comments with emoji feature and post-comments with emoji feature was +1.90% according to methods and +2.08% according to scenarios, as shown in Figure 8. The F1 improvement was favorable because it was higher than the accuracy. The algorithms that experienced the most significant improvement were RF and XGB. Unfortunately, the KNN7 got the worst improvement. RF had the most significant improvement in 1CPSMB and 2CPSMB. Figure 8 also shows negative values, particularly in KNN and BNB.

Based on comparative data on the effect of emojis on comments and post comments, it can be seen that the impact of emojis on comments or post-comments was quite good. Emojis improved spam comment detection performance compared to that was done without emoji features. The post-comment pair could still improve the performance using the horizontal stacked TF-IDF vectors approach. In general, the post-comment pair approach was also effective for all the emoji symbol scenarios that usually get a low result in the comment-only scenario.

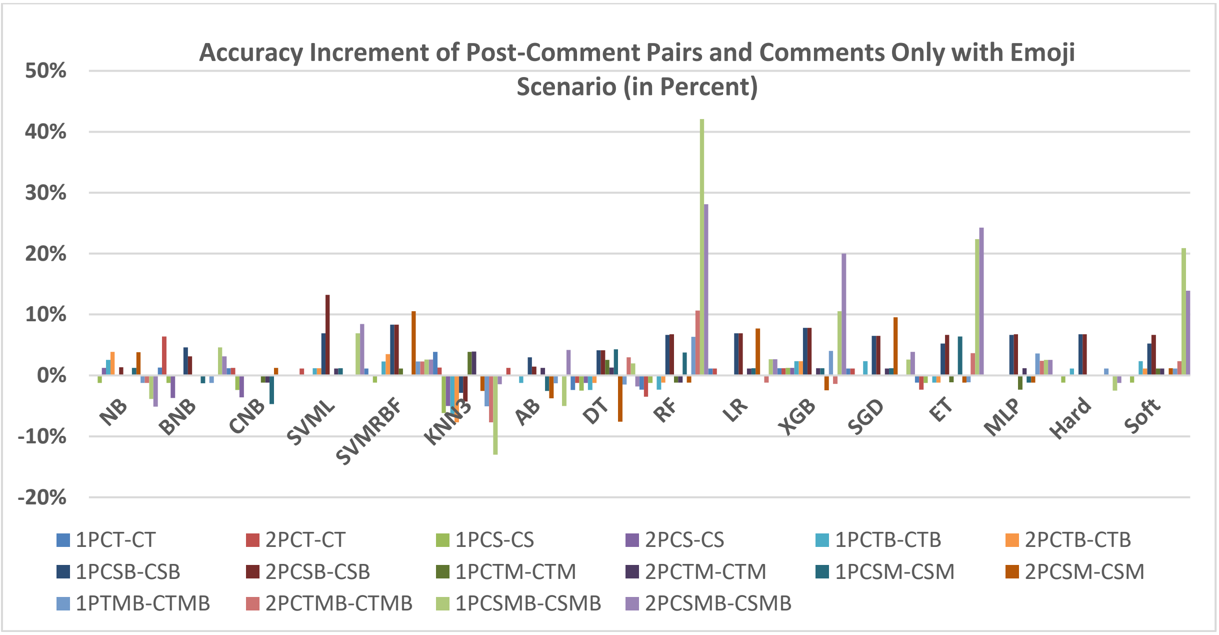


FIGURE 7. Accuracy Increment of Post-Comment Pairs and Comment Only (With Emoji) Scenario

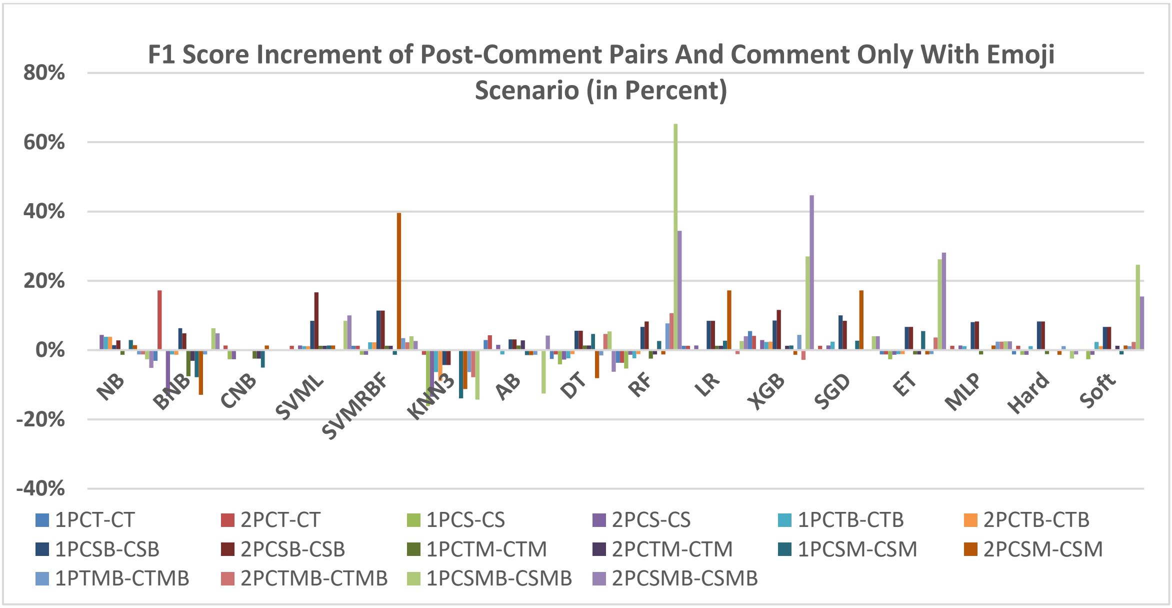


FIGURE 8. **F1 Score Comparison Between Comment Only Data and Post-Comment Pairs Data with Emoji (In Percent)**

8) PERFORMANCE COMPARISON BETWEEN EMOJI TEXT AND EMOJI SYMBOLS ON COMMENTS AND POST-COMMENT PAIRS

In this section, we compare the effect of converted emojis in text and symbols to get the best performance. Based on Figure 9, emoji text improved the average accuracy of comment-only data by 9.41% compared to emoji symbols. It can be stated that emoji text was better than emoji symbols because emoji symbols could not be learned quickly by using ML. Since there was no negative difference, it can be concluded that emoji text was superior to emoji symbols across all ML methods and scenarios. There was a drawback to this result. We had to convert emoji symbols to text before detecting spam comments. XGB and RF reached the most considerable average improvement. On the other hand, the lowest was the KNN7 method. The best method was XGB in 1CTSMB (1-gram comment manual features balanced). In contrast, KNN7 was the worst method in the 1CTS scenario.

Figure 10 shows the average improvement accuracy between emoji text and emoji symbols in post-comment pairs data was +6.98%, lower than the comment-only data. The highest average method was AB which reached a value of +33.33%, followed by XGB at +29.82%. The lowest average method was RF, with a value of 3.09%, higher than the lowest average method in comment-only data (+1.81%). The F1 score comparison between comment emoji text and comment emoji symbols had an average of 6.98%. However, the post-comment comparison got an average of 10.73%, which was higher than the accuracy. DT method got the highest average accuracy increment score. The F1 score comparison could not be displayed here due to the word-count limit of this article. Figures 9 and 10 illustrate the accuracy performance between emoji text and emoji symbols on comment-only data and post-comment pairs. The hard ensemble voting performed better in the accuracy and F1 score increment comparison.

We believe that post-comment pairs data promises further investigation because it allows for pairing post-context data with comments. The use of post-comment as a pair can provide the contextual relation between a post and a comment, so it can detect whether the comment is related or not to the post. In the end, we could determine whether a comment was spam by using the relation and the context.

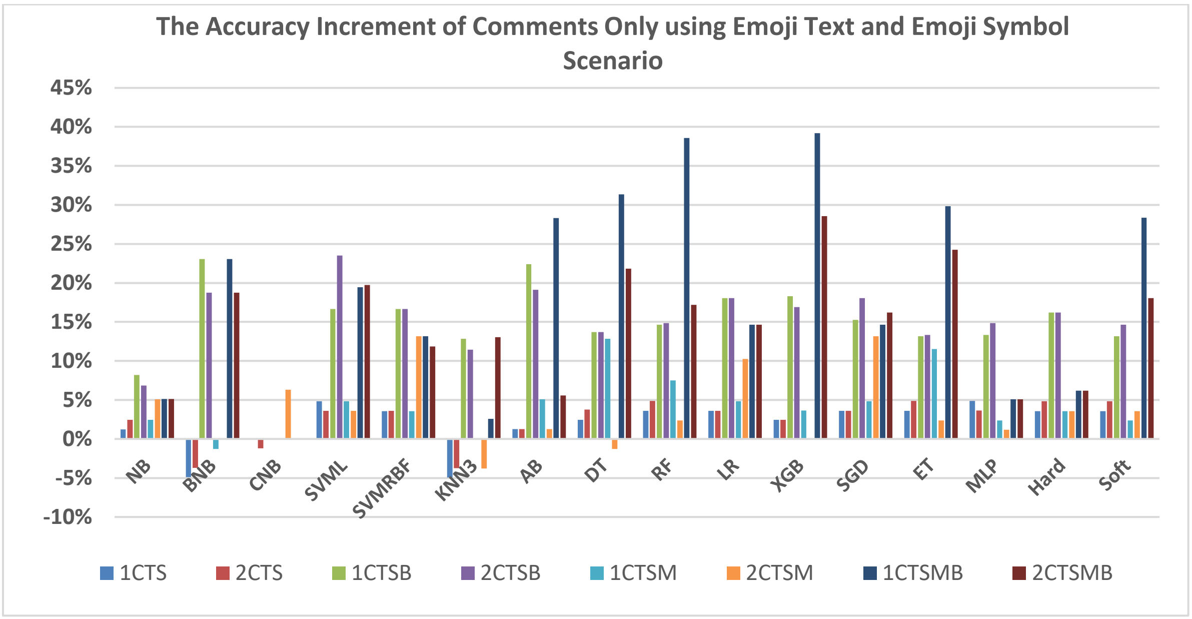


FIGURE 9. Accuracy Comparison Between Emoji Text and Emoji Symbol in Comment Only Data (In Percent)

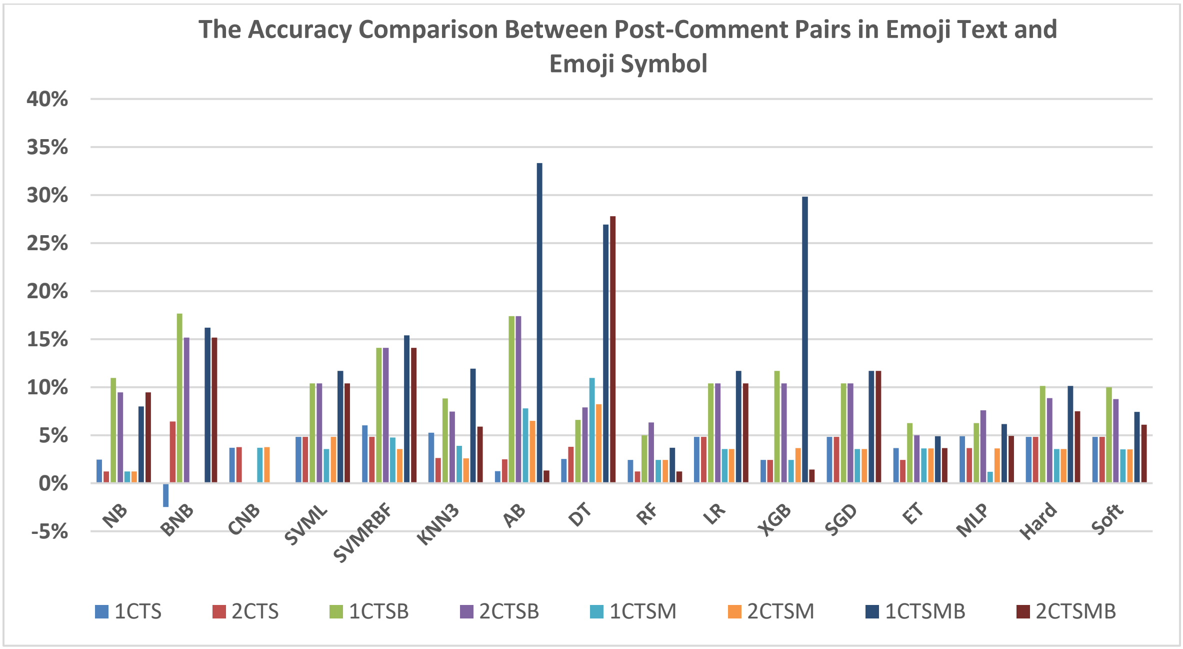


FIGURE 10. Accuracy Comparison Between Emoji Text and Emoji Symbol in Post-Comment Pairs (In Percent)

9) PERFORMANCE COMPARISON BETWEEN EMOJI POST-COMMENT PAIRS AND POST-COMMENT CONCATENATION APPROACH, MANUAL FEATURES, ENSEMBLE METHOD, AND BALANCED SCENARIO.

In the final section, we evaluate the comparative performance between post-comment pairs using two approaches. The first approach was using the post-and-comment data in TFIDF vectors and then stacking them horizontally as a pair vector. The second approach was using the post-and-comment data but by concatenating them as single sentences (post concatenated with comment) and then converting them into a TFIDF vector as a single vector. We also compared the impact of manual features and balanced/unbalanced dataset scenarios. Table XIV shows that the summary of the average accuracy improvement of post-comment as horizontally stacked pairs was +5.49% than post-comment concatenate (join string) with emojis. On the other hand, the use of post-comment as the concatenated string post-and-comment dropped to -6.97% even from the comment-only data in the average F1 score.

Moreover, the use of a concatenated string of posts and comments also dropped by -4.5% in average accuracy compared to post-comment stacked pairs. We can see that post-comment use in concatenated data was worse than that in horizontal stacked pairs data. We believe the horizontally stacked pairs of TFIDF post-comment vectors are one of the best approaches to represent the post-comment pairs data using ML techniques. Emojis had more significant features compared to those without emojis in comments only and post-comment. Emoji text is better than emoji symbols. Manual features and balanced scenarios also increased the accuracy and F1 score. The best scenario from all the experiments was the comment/post-comment emoji text to add the feature. Soft ensemble voting got the best average accuracy and F1 score compared to hard voting.

TABLE XIV. AVERAGE ACCURACY AND F1 SCORE INCREMENT OF POST-COMMENT PAIRS APPROACH AND POST-COMMENT CONCAT, MANUAL FEATURES, ENSEMBLE, AND BALANCED SCENARIO (IN PERCENT)

Scenario (with emoji)	Avg Accuracy Increment	Avg F1 Score Increment
Post-comment stacked pairs vs. post-comment concatenate	+5.49	+6.97
Comment-only vs. post-comment concatenate	+4.5	+5.88
Manual features addition vs. regular	+3,75	+1,89
Ensemble hard-voting post-comment pairs vs. comment only	+0.6	+0.55
Ensemble soft voting post-comment pairs vs. comment only	+3.1	+3.25
Balanced vs. unbalanced	+2.19	+2.96

10) ANALYSIS AND DISCUSSION

Based on our comprehensive study of many scenarios we discussed previously and the performance comparisons, it can be concluded that emojis significantly improved the detection performance of machine learning systems. Improved performance of emoji usage could reach an average of +4.65% to +6.64% in terms of accuracy and F1 score. Using post-comment as stacked pairs could improve the performance by about +5.49% to +6.97% rather than as a concatenated post-comment. Using emoji text was also better than emoji symbols in every scenario. Using manual features could increase the performance from +1.53% to +3.75% in accuracy. The ensemble methods could improve the performance from +0.6% to +3.25%. The balanced dataset also increased by +2.19% to +2.96%, better than the unbalanced dataset.

Emoji in text format performed better since the emoji symbol format was more difficult to process by pre-processing, and the sklearn's TF-IDF library uses word-based delimiters. Meanwhile, the pre-processing section and the TFIDF framework fully support emojis in text format. The dataset converted into a balanced dataset also improved the performance, particularly F1 scores, because the spam and non-spam categories became more proportional than before. The addition of manual features, such as in Table VB, could also improve the characteristics of the data so that it could be detected better.

Based on the data obtained, it can also be seen that the best methods capable of detecting spam comments were the SVM-RBF, RF, and ET. Most were occupied by tree-based algorithms, boosting, and ensemble learning. MLP as a primary deep learning method also yielded promising results, but it still needed to be explored further, especially pertinent to hyper-parameters and various other architectures. The detection performance value only reached an average between 74.1% and 84.56% in accuracy and between 71,4% and 81% in the F1 score.

The proposed ensemble machine learning with soft voting could achieve the best average in both accuracy and F1 score because the soft voting ensemble method could select the best classifier using the probability and threshold automatically. These ensemble methods can be used as the final model for the production mode. Hard voting had a lower performance because it used only the majority voting between the classifiers.

All the experiments attempted to use the comment dataset independently as a stand-alone dataset, as well as the post-and-comment datasets as horizontally stacked pair vectors. Merging post-comment data as concatenated data yielded poorer results than merging post-comment data as post-comment pairings. It was still necessary for remark spam detection to pay closer attention to the post context. Deep learning is an alternative technique that must be evaluated with exemplary architecture, especially for processing the context between comments and posts as a pair of input data that is simultaneously processed. Further research requires the detection of spam comments as an integral component of the document. A comment is regarded spam (irrelevant to post data) if the detection procedure is carried out in accordance with the context of the post. The process of spam detection will be investigated as a classification subtask known as sentence-pair classification.

V. CONCLUSION

This research aimed to enhance the detection of spam comments on social media with comprehensive experiments and analysis based on various test scenarios. This research differed from other studies that did not include the emoji feature in its detection method and only detected spam from the content of the comments. This study investigated the features of emojis and post-comment pair data to determine the optimal method, scenario, and features.

The experiment was conducted using 14 state-of-the-art ML models with various scenarios using the SpamID-Pair dataset to determine the significance of emoji features, which were usually ignored in many NLP types of research. We also investigated the use of post-comment pairs of TFIDF vectors stacked horizontally to enhance the performance. The results demonstrate the performance and comparison of accuracy and F1 scores across the various scenarios. The text emoji feature could enhance spam comment detection on social media, as evidenced by the performance improvement using machine learning methods by an average of 4% to 6%. Post-comment pairs data was also proven to improve detection performance by an average of 0.7% to 2.11%. To the best of our knowledge, this spam comment detection based on the post and comment as a pair is the first to conduct, especially in the context of Indonesian social media users. Adding manual features could also enhance detection performance by an average of 1.35% to 2.18%. The best methods for spam comment detection were SVM-RBF, RF, and ET algorithm using the C-PCTM and C-PCTMB scenarios. The ensemble soft voting method yielded the best average performance in both accuracy and F1 score rather than a single classifier. It could be used in production mode. However, it has one disadvantage due to its big-size model compared to each/single model without the ensemble technique. In conclusion, using emojis, a post-comment pairs approach, and balanced-manual features in both comments and pairs of comments did improve the performance.

However, this research may not yet fully understand the context between posts and comments using machine learning. A suitable model and method to determine the semantic relationship are still required in future studies. The context between posts and comments is crucial to know the relevance between comments and posts, so spam comments can be better detected to increase the accuracy and F1 score. We intend to apply the deep learning model in sentence pairs classification adaptation [49] and adjustment between post and comment vector representations to determine their relevance. The comment that is not relevant to the post tends to be spam.

ACKNOWLEDGMENT

This research is supported by the Department of Computer Science and Electronics with Grant No. 241/J01.1.28/PL.06.02/2022. We want to thank the Department of Computer Science and Electronics, Universitas Gadjah Mada, and the Faculty of Information Technology, Universitas Kristen Duta Wacana, for all the support.

REFERENCES

[1] Databooks, “Ini Media Sosial Paling Populer Sepanjang April 2020,” *Databooks*, 2020. https://databoks.katadata.co.id/datapublish/2020/05/25/ini-media-sosial-paling-populer-sepanjang-april-2020 (accessed Nov. 04, 2020).
[2] S. Aiyar and N. P. Shetty, “N-Gram Assisted Youtube Spam Comment Detection,” *Procedia Comput. Sci.*, vol. 132, pp. 174–182, 2018, doi: 10.1016/j.procs.2018.05.181.
[3] A. R. Chrismanto, A. K. Sari, and Y. Suyanto, “CRITICAL EVALUATION ON SPAM CONTENT DETECTION IN SOCIAL MEDIA,” *J. Theor. Appl. Inf. Technol.*, vol. 100, no. 8, pp. 2642–2667, 2022, [Online]. Available: http://www.jatit.org/volumes/Vol100No8/29Vol100No8.pdf
[4] W. Zhang and H.-M. Sun, “Instagram Spam Detection,” in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, IEEE, Jan. 2017, pp. 227–228. doi: 10.1109/PRDC.2017.43.
[5] B. Priyoko and A. Yaqin, “Implementation of naive bayes algorithm for spam comments classification on Instagram,” in *2019 International Conference on Information and Communications Technology, ICOIACT 2019*, IEEE, 2019, pp. 508–513. doi: 10.1109/ICOIACT46704.2019.8938575.
[6] N. A. Haqimi, N. Rokhman, and S. Priyanta, “Detection Of Spam Comments On Instagram Using Complementary Naïve Bayes,” *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 13, no. 3, p. 263, Jul. 2019, doi: 10.22146/ijccs.47046.
[7] A. Chrismanto and Y. Lukito, “Identifikasi Komentar Spam Pada Instagram,” *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 8, no. 3, p. 219, 2017, doi: 10.24843/lkjiti.2017.v08.i03.p08.
[8] A. Chrismanto, Y. Lukito, and A. Susilo, “Implementasi Distance Weighted K-Nearest Neighbor Untuk Klasifikasi Spam dan Non-Spam Pada Komentar Instagram,” *J. Edukasi dan Penelit. Inform.*, vol. 6, no. 2, p. 236, 2020, doi: 10.26418/jp.v6i2.39996.
[9] F. Prabowo and A. Purwarianti, “Instagram online shop’s comment classification using statistical approach,” in *Proceedings - 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering, ICITISEE 2017*, Yogyakarta: IEEE, 2018, pp. 282–287. doi: 10.1109/ICITISEE.2017.8285512.
[10] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, “An up-to-date comparison of state-of-the-art classification algorithms,” *Expert Syst. Appl.*, vol. 82, pp. 128–150, 2017, doi: 10.1016/j.eswa.2017.04.003.
[11] C. Mus, “10+ Akun Instagram Dengan Followers Terbanyak Di Indonesia,” *musdeoranje.net*, 2015. http://www.musdeoranje.net/2016/08/akun-instagram-dengan-followers-terbanyak-di-indonesia.html (accessed Oct. 13, 2021).
[12] S. Rao, A. K. Verma, and T. Bhatia, “A review on social spam detection: Challenges, open issues, and future directions,” *Expert Syst. Appl.*, vol. 186, no. March, p. 115742, 2021, doi: 10.1016/j.eswa.2021.115742.
[13] P. K. Roy, J. P. Singh, and S. Banerjee, “Deep learning to filter SMS Spam,” *Futur. Gener. Comput. Syst.*, vol. 102, pp. 524–533, 2020, doi: 10.1016/j.future.2019.09.001.
[14] A. Chandra and S. K. Khatri, “Spam SMS Filtering using Recurrent Neural Network and Long Short Term Memory,” *2019 4th Int. Conf. Inf. Syst. Comput. Networks, ISCON 2019*, pp. 118–122, 2019, doi: 10.1109/ISCON47742.2019.9036269.
[15] A. A. Septiandri and O. Wibisono, “Detecting spam comments on Indonesia’s Instagram posts,” *J. Phys. Conf. Ser.*, vol. 801, no. 012069, pp. 1–7, 2017, doi: 10.1088/1742-6596/755/1/011001.
[16] A. Chrismanto and Y. Lukito, “Klasifikasi Komentar Spam Pada Instagram Berbahasa Indonesia Menggunakan K-NN,” in *Seminar Nasional Teknologi Informasi Kesehatan (SNATIK)*, Yogyakarta: STIKES Surya Global, 2017, pp. 298–306.
[17] A. Talha and R. Kara, “A Survey of Spam Detection Methods on Twitter,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 3, pp. 29–38, 2017, doi: 10.14569/ijacsa.2017.080305.
[18] N. M. Samsudin, C. F. B. Mohd Foozy, N. Alias, P. Shamala, N. F. Othman, and W. I. S. Wan Din, “Youtube spam detection framework using naïve bayes and logistic regression,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 14, no. 3, pp. 1508–1517, 2019, doi: 10.11591/ijeecs.v14.i3.pp1508-1517.
[19] N. Alias, C. F. M. Foozy, and S. N. Ramli, “Video spam comment features selection using machine learning techniques,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 15, no. 2, pp. 1046–1053, 2019, doi: 10.11591/ijeecs.v15.i2.pp1046-1053.
[20] N. Banik and M. H. H. Rahman, “Toxicity Detection on Bengali Social Media Comments using Supervised Models,” *ICIET 2019 - 2nd Int. Conf. Innov. Eng. Technol.*, pp. 23–24, 2019, doi: 10.1109/ICIET48527.2019.9290710.
[21] R. Abinaya, E. Bertilla Niveda, and P. Naveen, “Spam detection on social media platforms,” *2020 7th Int. Conf. Smart Struct. Syst. ICSSS 2020*, pp. 31–33, 2020, doi:

10.1109/ICSSS49621.2020.9201948.

[22] P. Shil, U. S. Rahman, M. Rahman, and M. S. Islam, "An Approach for Detecting Bangla Spam Comments on Facebook," *Proc. Int. Conf. Electron. Commun. Inf. Technol. ICECIT 2021*, no. September, pp. 14–16, 2021, doi: 10.1109/ICECIT54077.2021.9641358.

[23] C. Fatichah, W. F. Lazuardi, D. A. Navastara, N. Suciati, and A. Munif, "A Content Filtering from Spam Posts on Social Media using Weighted Multimodal Approach," *J. Comput. Sci.*, vol. 17, no. 1, pp. 55–66, 2021, doi: 10.3844/jcssp.2021.55.66.

[24] H. Oh, "A YouTube Spam Comments Detection Scheme Using Cascaded Ensemble Machine Learning Model," *IEEE Access*, pp. 144121–144128, 2021, doi: 10.1109/ACCESS.2021.3121508.

[25] Y. Tashtoush, A. Magableh, O. Darwish, L. Smadi, O. Alomari, and A. ALghazoo, "Detecting Arabic YouTube Spam Using Data Mining Techniques," in *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, IEEE, Jun. 2022, pp. 1–5. doi: 10.1109/ISDFS55398.2022.9800840.

[26] A. Sinhal and M. Maheshwari, "YouTube: Spam Comments Filtration using Hybrid Ensemble Machine Learning Models," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 12, no. 10, pp. 169–182, 2022, doi: 10.46338/ijetae1022_18.

[27] R. Wongso, F. A. Luwinda, B. C. Trisnajaya, O. Rusli, and Rudy, "News Article Text Classification in Indonesian Language," *Procedia Comput. Sci.*, vol. 116, pp. 137–143, 2017, doi: 10.1016/j.procs.2017.10.039.

[28] F. Z. Ruskanda, "Study on the Effect of Preprocessing Methods for Spam Email Detection," *Indones. J. Comput.*, vol. 4, no. 1, p. 109, 2019, doi: 10.21108/indojc.2019.4.1.284.

[29] W. Etaiwi and G. Naymat, "The Impact of applying Different Preprocessing Steps on Review Spam Detection," *Procedia Comput. Sci.*, vol. 113, pp. 273–279, 2017, doi: 10.1016/j.procs.2017.08.368.

[30] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, Jan. 1988, doi: 10.1016/0306-4573(88)90021-0.

[31] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*, 1st editio. Cambridge: Cambridge University Press, 2008. doi: 10.1017/cbo9780511809071.

[32] H. Zhang, "The Optimality of Naive Bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, Florida, USA: AAAI Press, 2004, pp. 562–567. [Online]. Available: http://www.aaai.org/Library/FLAIRS/2004/flairs04-097.php

[33] Suyanto, *Machine Learning Tingkat Dasar dan Lanjut*, 1st ed. Bandung: Penerbit Informatika, 2018.

[34] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," in *ICML'03: Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, Washington, DC, USA: AAAI Press, 2003, pp. 616–623. doi: 10.5555/3041838.3041916.

[35] Scikit-Learn, "1.4. Support Vector Machines — scikit-learn 0.23.2 documentation," *Scikit-Learn Documentation*, 2021. https://scikit-learn.org/stable/modules/svm.html (accessed Nov. 19, 2020).

[36] Suyanto;, *Data mining untuk klasifikasi dan klasterisasi data*, 1st ed. Bandung: Informatika, 2017. Accessed: Nov. 19, 2020. [Online]. Available: //catalogue.ubharajaya.ac.id/slims/index.php?p=show_detail&id=39879

[37] J. Han, M. Kamber, and J. Pei, *Data Mining : Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011. Accessed: Nov. 19, 2020. [Online]. Available: https://www.amazon.com/Data-Mining-Concepts-Techniques-Management/dp/0123814790

[38] P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 647–648, 2001, doi: 10.1109/icdm.2001.989592.

[39] J. R. Quinlan, "Induction of Decision Trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986, doi: 10.1023/A:1022643204877.

[40] Sklearn, "sklearn.tree.DecisionTreeClassifier," *sklearn 1.1.1 documentiation*, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (accessed Jul. 24, 2022).

[41] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997, doi: 10.1006/jcss.1997.1504.

[42] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[43] N. Bhandari, "A Gentle Introduction to XGBoost for Applied Machine Learning," *Medium*, 2018. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed Dec. 16, 2020).

[44] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach Learn*, vol. 63, pp. 3–42, 2006, doi: 10.1007/s10994-006-6226-1.

[45] J. Brownlee, "ExtraTreesClassifier. How does ExtraTreesClassifier reduce… | by Naman Bhandari | Medium," *Machine Learning Mastery*, 2016. https://medium.com/@namanbhandari/extratreesclassifier-8e7fc0502c7 (accessed Dec. 16, 2020).

[46] L. Rokach, *Pattern Classification Using Ensemble Methods*, vol. 75. in Series in Machine Perception and Artificial Intelligence, vol. 75. WORLD SCIENTIFIC, 2009. doi: 10.1142/7238.

[47] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informatics*, vol. 17, no. 1, pp. 168–192, 2018, doi: 10.1016/J.ACI.2018.08.003/FULL/PDF.

[48] A. R. Chrismanto, A. K. Sari, and Y. Suyanto, "SPAMID-PAIR: A Novel Indonesian Post–Comment Pairs Dataset Containing Emoji," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 11, pp. 92–100, 2022, doi: 10.14569/IJACSA.2022.0131110.

[49] J. Pei, Y. Wu, Z. Qin, Y. Cong, and J. Guan, "Attention-based model for predicting question relatedness on Stack Overflow," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2021, pp. 97–107. doi: 10.1109/MSR52588.2021.00023.

ANTONIUS RACHMAT CHRISMANTO Antonius Rachmat Chrismanto, S.Kom., M.Cs. has been a senior lecturer and doctoral student at Universitas Gadjah Mada since 2020. His research interests are text mining, natural language processing, and social media analysis. He got his bachelor's degree from Universitas Kristen Duta Wacana, Indonesia (2004), and his master's degree from Universitas Gadjah Mada, Indonesia (2008). He also authored two books on algorithms and GUI programming. His publications are available on Research Gate.

ANNY KARTIKA SARI Anny Kartika Sari, S.Si., M.Sc., Ph.D., is a senior lecturer and associate professor at Universitas Gadjah Mada. Her research interests are discrete structures and ontology. She got her bachelor's degree from Universitas Gadjah Mada (2000), master's degree from Universiteit Twente, The Netherlands (2004), and Ph.D. from La Trobe University, Australia (2014).

YOHANES SUYANTO Dr. Yohanes Suyanto, M.Ikom, is a senior lecturer and associate professor at Universitas Gadjah Mada. His research interests are text-to-speech, multimedia, and GIS. He got her bachelor's degree from Universitas Gadjah Mada (1987), a master's degree from Universitas Indonesia, and a Doctoral degree from Universitas Gadjah Mada (2014).