Vol. 4 No. (1) (June 2026) 27-42

Journal of Data Insights

http://journalnew.unimus.ac.id/index.php/jodi

Comparison of Decision Tree and Random Forest for Sentiment Analysis of Info BMKG Mobile App Reviews

Aditya Rizky Purnama[1], Galet Guntoro Setiaji[2], Ahmad Rifa'i[3]
[1,2,3]Faculty of Information and Communication Technology, Universitas Semarang, Indonesia

DOI: https://doi.org/10.26714/jodi.v4i1.1112

Article Info

Article History:
Submitted May 04, 2026
Revised June 02, 2026
Approved June 03, 2026

Keywords:
BMKG application; decision tree; random forest; sentiment analysis; TF-IDF.

Abstract

The Info BMKG app is a digital public service platform developed by the government to provide real-time weather, earthquake, and climate information to the Indonesian public. The large number of user reviews available on the Google Play Store holds great potential as a source for service evaluation. However, the limitations of manual analysis make a computational approach necessary so that the review processing can be carried out more efficiently and in a structured manner. This study proposes a machine learning-based sentiment analysis framework to classify user reviews of the Info BMKG app, while comparing the performance of the Decision Tree and Random Forest algorithms using 10,000 review data points collected via web scraping techniques. The data underwent text preprocessing, sentiment labeling based on ratings, TF-IDF feature extraction, class imbalance handling using balanced class weighting, and hyperparameter optimization using GridSearchCV and RandomizedSearchCV. Evaluation was conducted using the metrics accuracy, precision, recall, F1-score, macro-averaged F1-score, cross-validation, and computation time. Test results show that Random Forest achieved an accuracy of 78% and a macro-averaged F1-score of 53%, outperforming Decision Tree, which achieved an accuracy of 73% and a macro-averaged F1-score of 52%. In terms of computational efficiency, the Decision Tree had a faster testing time of 0.0991 seconds compared to the Random Forest's 0.1999 seconds, while the Random Forest had a faster training time of 2.7800 seconds compared to the Decision Tree's 7.1717 seconds. These findings indicate that Random Forest is the more optimal algorithm for classifying the sentiment of public service app reviews because it produces better classification performance, although the Decision Tree still has an advantage in prediction speed.

✉ Correspondence Address:
E-mail: kikyadityaa@gmail.com

e-ISSN: 2988 - 2109

INTRODUCTION

Advances in information technology over the past decade have transformed the way people consume information. One tangible manifestation of this change is the proliferation of public service-based mobile apps capable of delivering data quickly. In the context of disaster management, the Indonesian government has developed the Info BMKG app as a platform for providing real-time data on weather conditions, seismic activity, and the national climate. Given Indonesia's geographical location, the existence of such an information platform plays a strategic role in mitigating natural disaster risks and supporting public safety preparedness [1]. As the user base of the Info BMKG app grows, the volume of comments submitted through distribution channels like the Google Play Store has also seen significant growth. These reviews contain a variety of user expressions ranging from positive, negative, to neutral which collectively reflect users' direct experiences interacting with the app and their overall satisfaction toward public digital services [2]. For the development team, this dataset is actually a valuable source of feedback to support service quality evaluation and improvement. However, the sheer volume of review data makes a manual analysis approach no longer sufficient, both in terms of time efficiency and the objectivity of the assessment due to human fatigue and cognitive bias in interpreting large-scale unstructured text [3]. This situation necessitates a computational solution capable of processing this data automatically and in a structured manner.

Sentiment analysis is a Natural Language Processing (NLP) approach used to identify opinions in text. This method is widely applied across various research fields. In this context, sentiment analysis has proven effective in revealing the distribution of user perceptions while providing constructive feedback to developers [1]. Similar studies on other public service applications also show that classification methods can provide an adequate picture of user satisfaction levels [4]. Furthermore, the application of sentiment analysis to digital wallet apps shows that the Random Forest algorithm is capable of achieving high classification accuracy for reviews across various platforms [3]. Findings from the MyPertamina app also underscore the relevance of Random Forest, SVM, and Naïve Bayes in large-scale data scenarios with imbalanced class distributions [5].

In the field of machine learning, various algorithms have been tested for sentiment analysis, including Naïve Bayes, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree, and Random Forest. Each algorithm has its own characteristics and strengths, with performance varying depending on the data characteristics and the approach used. From various comparative studies that have been conducted, Random Forest has consistently demonstrated superiority, achieving the highest accuracy of 95.53% and outperforming other algorithms like Decision Tree, SVM, and KNN [6]. On the other hand, decision trees are known for their high interpretability, although they are prone to overfitting under certain data conditions. Wulandari's research on Canva app reviews in the Play Store showed that decision trees achieved an accuracy of 87% with the support of a lexicon-based labeling approach, confirming that this algorithm remains competitive for sentiment classification tasks [7]. In the context of public service applications, high interpretability is crucial as it allows developers to easily trace the explicit logic behind user complaints and satisfaction drivers.

A comparative study of Random Forest and Decision Trees in the context of sentiment analysis shows that Random Forest tends to deliver more consistent performance, thanks to its nature as an ensemble method that integrates multiple decision trees [8]. Nevertheless, decision trees remain a relevant point of comparison because they offer computational efficiency and a simple model structure. Both algorithms continue to be the subject of study in recent comparative research [9]. The pairing of these two algorithms serves as an ideal benchmark for public service application reviews, which are notoriously prone to high noise, informal language, and extreme class imbalances. While the Decision Tree provides a baseline transparent model to capture direct patterns, Random Forest acts as a robust mechanism to stabilize predictions against the unstructured nature of citizen feedback. From the perspective of addressing class imbalance, Setiaji's research reveals that applying the undersampling method with random state parameters in Random Forest can significantly improve

average accuracy, reinforcing the argument that appropriate parameter configurations have a major impact on model performance [10]. In addition, Indriani's study on the GoPay app reviews showed that combining Random Forest with TF-IDF and SMOTE optimization yielded a remarkable accuracy of 90.00%, outperforming both SVM and Decision Tree, thereby underscoring the importance of preprocessing and model selection on classification quality [11].

Although the existing literature has made significant contributions, a number of research gaps remain. Most previous studies have relied on small-scale datasets or limited their testing to a single algorithm. Research related to the BMKG Info application has generally focused on algorithms such as Naïve Bayes, SVM, and KNN, so studies specifically comparing Random Forest and Decision Tree using larger and more up-to-date datasets remain very limited. This situation underscores the urgency for further research with more representative data coverage and a more comprehensive comparative approach.

This study aims to analyze the sentiment of BMKG Info reviews using the Decision Tree and Random Forest algorithms to compare their performance. The research process includes data collection via web scraping, preprocessing, labeling, TF-IDF feature extraction, as well as model classification and evaluation. The main contribution of this research is the availability of a comparative study of algorithms on the Info BMKG dataset, while also providing recommendations for an optimal model as a reference for future sentiment analysis studies.

METHOD

2.1 Research Process

This study was conducted through several sequential and interrelated stages. Overall, the research process began with the collection of user review data for the Info BMKG app from the Google Play Store using web scraping techniques, followed by preprocessing to ensure the data was clean and ready for analysis. The subsequent steps include automatic sentiment labeling based on rating values, transforming text data into numerical representations using the TF-IDF method, splitting the dataset, training models using the Decision Tree and Random Forest algorithms, and evaluating and analyzing the performance comparison of the two models. An overview of the research workflow is presented in Figure 1.

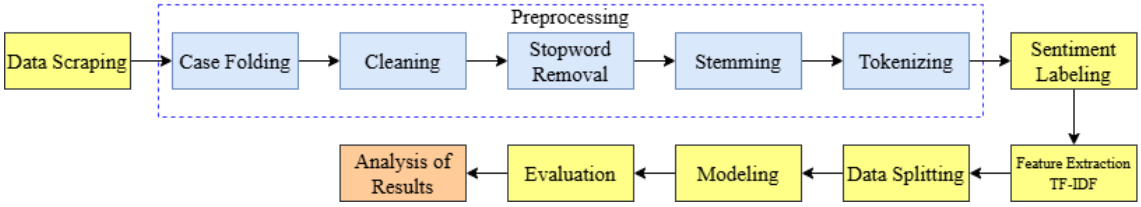

Figure 1. Research Process

2.2 Data Collection

The data used is sourced from user reviews of the Info BMKG app, which are publicly available on the Google Play Store. The data was collected using a web scraping technique with the `google-play-scraper` library in Python. This technique enables automatic data acquisition without the need for manual entry [12]. The scraping process successfully collected 10,000 reviews published between November 2019 and April 2026, each containing the reviewer's name, rating, date, and review content. The collected data was then filtered so that only the attributes relevant to the analysis were included in the subsequent stages.

2.3 Text Preprocessing

This stage is implemented to prepare the review data so that it can be processed optimally by the machine learning algorithm. This is because raw data typically contains symbols and numbers that

are irrelevant to the analysis [13]. In this study, preprocessing was carried out in five sequential steps as follows:

a. Case Folding: convert all characters to lowercase.
b. Cleaning: the process of removing numbers, punctuation marks, special characters such as emojis, and extra spaces.
c. Stopword Removal: the process of removing functional words with minimal semantic meaning.
d. Stemming: converting derived words to their root forms.
e. Tokenizing: breaking down the cleaned text into individual word units, which are stored in the `ulasan_tokenized` column.

The comprehensive application of preprocessing steps has been shown to improve the quality of classification models by reducing noise in the text data [14].

2.4 Sentiment Labeling

The sentiment labeling process is based on the star ratings submitted by users. These are then grouped into three sentiment categories according to the following rules:

a. Positive: Scores of 4 and 5.
b. Neutral: Score 3.
c. Negative: Scores of 1 and 2.

This rating-based approach was chosen because the star ratings given by users are seen as an explicit representation of their level of satisfaction with the app [15].

2.5 Feature Extraction (TF-IDF)

To convert text data into a vector representation that can be understood by machine learning models, this study employs the TF-IDF method as a vectorization technique applied after the preprocessing stage is complete. This method relies on assigning a weight to each word, which is determined by the word's degree of significance within a document relative to its distribution across the entire corpus [16]. Mathematically, this method consists of two calculation components, namely:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (1)$$

$$IDF(t) = log\left(\frac{N}{df_t}\right) \quad (2)$$

$$TFIDF(t,d) = TF(t,d) \times IDF(t) \quad (3)$$

Formula Description:

$TF(t,d)$: The frequency of the letter t in the review document d.
$IDF(t)$: The rarity or importance of the word t across the entire document.
N : Total number of review documents in the corpus.
df_t : The number of documents in the corpus that contain the word t.
$TFIDF(t,d)$: The final weight of the word t in document d from the multiplication result

The application of this method is effective in mitigating the influence of words that appear frequently but have low informational relevance in the context of classification.

2.6 Data Splitting

Next, the data is divided into two main sets training and testing in an 80:20 ratio. This ratio is set so that the model receives a portion of the data for training while still retaining data to test its ability to generalize to previously unseen data [17].

2.7 Modeling

The sentiment classification process for reviews of the Info BMKG app was conducted by comparing two tree-based algorithms: Decision Tree and Random Forest. To produce a model with optimal performance and ensure the reproducibility of this study, hyperparameter tuning was explicitly performed using training data via cross-validation (with CV = 3). To address the issue of

data imbalance (class imbalance), both models were configured with balanced class weights (class_weight='balanced').

2.7.1 Decision Tree

Parameter optimization for the Decision Tree algorithm was performed using the Grid Search method (GridSearchCV) by evaluating combinations of tree depth (max_depth) and the minimum number of samples required for a node split (min_samples_split). Based on the tuning results, the final model is configured using the Gini Impurity splitting criterion with the maximum depth (max_depth) parameter set to None so that the tree can grow fully without restrictions to capture the complexity of TF-IDF features comprehensively. Additionally, the minimum sample threshold for node splitting (min_samples_split) was set to 5, and the random_state was set to 42 to control randomness and ensure consistency in model replication results.

Mathematically, the determination of feature splitting (splitting nodes) at each sequential stage of the tree is calculated using the Gini Impurity criterion (Gini(D)), which measures the level of impurity of a data node (D). The use of this criterion is consistent with previous research that has demonstrated the effectiveness of tree-structure-based node splitting for text data [6]. The Gini Impurity formula is defined as follows:

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{4}$$

Formula Description:

$\text{Gini}(D)$: The Gini impurity of the dataset or node D.
c : The total number of target classes
p_i : The proportion of a review belonging to class i appearing at that node.

2.7.2 Random Forest

Optimization of the Random Forest algorithm was performed using the Randomized Search approach (RandomizedSearchCV), with the search space encompassing the number of tree estimators (n_estimators), the maximum depth, and the sample size limit for node splitting. Based on the results of 5 iterations of random search (n_iter=5), the final ensemble model was configured with 150 independent decision trees built in parallel. To avoid the weakness of single trees, which are prone to overfitting on complex data structures, the maximum depth (max_depth) of each tree was limited to level 30. Furthermore, the minimum number of samples for node splitting (min_samples_split) is set to 10, with the randomness control set to random_state=42 to ensure consistent model performance when replicated.

Each tree within this Random Forest structure is trained using a random subsample of the training data via the bagging method. This ensemble approach has proven effective in minimizing variance and improving the consistency of classification accuracy in complex text data cases [17]. The final prediction of this ensemble model is determined through a majority voting mechanism based on the individual predictions of all decision trees. The mathematical formulation of the Random Forest final prediction function is written as follows:

$$\hat{y} = mode\{h_1(x), h_2(x), \dots, h_B(x)\} \tag{5}$$

Formula Explanation:

$\hat{y}$: The final predicted sentiment label for a review data point x.
mode : The class with the highest frequency.
$h_b(x)$: The prediction result for the b node of the tree based on the input x.
B : The total number of tree estimators ($B = 150$).

2.8 Model Evaluation

The performance of each model was evaluated using a number of standard evaluation metrics, including:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$\text{Macro Precision} = \frac{P_{\text{negatif}} + P_{\text{netral}} + P_{\text{positif}}}{3} \quad (7)$$

$$\text{Macro Recall} = \frac{R_{\text{negatif}} + R_{\text{netral}} + R_{\text{positif}}}{3} \quad (8)$$

$$\text{Macro F1-Score} = \frac{F1_{\text{negatif}} + F1_{\text{netral}} + F1_{\text{positif}}}{3} \quad (9)$$

$$\text{Macro Average} = \frac{1}{K}\sum_{i=1}^{K} M_i \quad (10)$$

The performance of each classification model was comprehensively evaluated using a combination of the Accuracy metric and the Macro-Averaged approach to ensure that the assessment was not biased toward the dominance of any single sentiment class.

The Accuracy metric is used to measure the percentage or proportion of correct predictions generated by the model relative to the entire test dataset, without distinguishing between label categories.

Meanwhile, the Macro Precision metric is calculated by summing the Precision values for the negative, neutral, and positive classes separately, which is then averaged by the total number of those three classes to assess the model's objectivity in predicting each sentiment.

To measure the model's ability to retrieve all original information from each category, the Macro Recall metric is used, which integrates the total recall from the three sentiment classes and divides it equally.

As the primary performance indicator in this study, the Macro F1-Score metric evaluates the average of the individual F1-Scores obtained from the negative, neutral, and positive classes. This metric is the most crucial parameter because it assigns equal weight to the performance of each review category even when there is an imbalance in the number of data samples across classes (class imbalance).

In general, all these macro calculations rely on the Macro Average formulation, which acts as a mathematical generalization scheme, where the values of specific evaluation metrics for each category are cumulatively summed and then divided by a divisor constant representing the total number of all target class variations.

In addition to the four metrics mentioned above, the evaluation also includes visualizations of the confusion matrix, ROC curve, and cross-validation. The use of this combination of evaluation metrics aims to provide a more comprehensive assessment of model performance that is not biased toward any particular class distribution [18].

2.9 Analysis of Results

The final stage of this research process is a comparative analysis of the performance of the two algorithms. The comparison was conducted comprehensively, taking into account evaluation metrics, computational efficiency, and model stability as measured through cross-validation. These findings are expected to provide objective recommendations regarding the most optimal algorithm for this study.

RESULTS AND DISCUSSION

3.1 Results

A series of experiments were conducted to evaluate the model's performance in sentiment classification. This study encompasses data collection, preprocessing, feature extraction, training of the Decision and Random Forest algorithms, as well as the evaluation and comparison of the prediction results from both models. This section provides a comprehensive overview of the results obtained at each stage.

3.1.1 Data Collection

The first step in this study was to collect review data using web scraping techniques. The scraping process was carried out using the Python programming language and libraries within the Jupyter Notebook environment. Data was retrieved using the parameters `lang="id"` and `country="id"` to ensure that the reviews obtained were in Indonesian and from domestic users, and were sorted by most recent (Sort.NEWEST).

Data retrieval is performed in batches (pagination), fetching 500 reviews per request using the continuation_token mechanism. To prevent the server from being overloaded, a one-second delay (time.sleep(1)) is applied each time a batch is retrieved. The scraping process will stop automatically when the token runs out or the total amount of data reaches the target of 10,000.

The data was then filtered into four main columns: username (userName), review date (at), star rating (score), and review content (content). The scraping results are shown in Figure 2, which displays the top 10 entries from the total number of reviews successfully collected.

	User Name	Date	Rating	Review
0	Aqiqah Malang	2026-04-21 11:53:49	4	good
1	Dian Bayu Nurhadi	2026-04-14 18:40:56	1	the application cannot be opened
2	Rudi Kussoy	2026-04-14 15:22:43	5	congratulations
3	Ambang Haryono	2026-04-13 17:50:28	5	This application is very good and makes it possible to know if there is an earthquake
4	bonzae kecil	2026-04-12 15:32:37	5	good, quite helpful
5	Rizal Linda	2026-04-11 20:02:34	5	awesome
6	JsstTonii	2026-04-11 14:11:01	1	Previously the application could still be opened although the loading was very long. now the application cannot be accessed at all. please fix it
7	Bula Yusuf	2026-04-11 00:32:28	3	good, but sometimes the system cannot be opened, please fix it again
8	mamat gratis	2026-04-10 14:27:11	5	very helpful for weather forecasting in my place
9	Nur Hidayati	2026-04-10 13:48:11	3	a lot of information is inaccurate, here it is raining heavily but the info on the app is thick cloudy

Figure 2. Data scraped from the Google Play Store

3.1.2 Text Preprocessing

Next, the data is cleaned and standardized so that it can be processed optimally by the algorithm. Preprocessing is performed using Python, utilizing the NLTK library for tokenization and stopword removal, as well as the Sastrawi library for Indonesian stemming.

The results of all preprocessing steps are stored in new columns named ulasan_cleaned and ulasan_tokenized. After preprocessing, data containing empty text were removed, leaving a final dataset of 9,552 reviews out of the total 10,000 data points successfully scraped. The preprocessing results are shown in Figure 3.

	User Name	Review	Cleaned Review	Tokenized Review
0	Aqiqah Malang	good	good	['good']
1	Dian Bayu Nurhadi	the application cannot be opened	application open	['application', 'open']
2	Rudi Kussoy	congratulations	congratulations	['congratulations']
3	Ambang Haryono	This application is very good and makes it possible to know if there is an earthquake 😊❤️🔥	application good if earthquake	['application', 'good', 'if', 'earthquake']
4	bonzae kecil	good, quite helpful	good help	['good', 'help']
5	Rizal Linda	awesome	awesome	['awesome']
6	JsstTonii	Previously the application could still be opened although the loading was very long. now the application cannot be accessed at all. please fix it 🙏	application open loading application access please good	['application', 'open', 'loading', 'application', 'access', 'please', 'good']
7	Bula Yusuf	good, but sometimes the system cannot be opened, please fix it again	good sometimes system cannot open please good	['good', 'sometimes', 'system', 'cannot', 'open', 'please', 'good']
8	mamat gratis	very helpful for weather forecasting in my place	help guess weather	['help', 'guess', 'weather']
9	Nur Hidayati	a lot of information is inaccurate, here it is raining heavily but the info on the app is thick cloudy	inaccurate info rain heavy info app cloudy thick	['inaccurate', 'info', 'rain', 'heavy', 'info', 'app', 'cloudy', 'thick']

Figure 3. Data from the Preprocessing of Reviews

3.1.3 Sentiment Labeling

Labeling based on ratings was chosen because it provides an explicit representation of user satisfaction that is already available in the scraped data, thereby eliminating the need for manual labeling. The labeling rules applied are as follows: reviews with ratings of 1 and 2 are categorized as negative, a rating of 3 is categorized as neutral, and ratings of 4 and 5 are categorized as positive. The results of the sentiment labeling are shown in Figure 4.

	User Name	Rating	Sentiment
0	Aqiqah Malang	4	Positive
1	Dian Bayu Nurhadi	1	Negative
2	Rudi Kussoy	5	Positive
3	Ambang Haryono	5	Positive
4	bonzae kecil	5	Positive
5	Rizal Linda	5	Positive
6	JsstTonii	1	Negative
7	Bula Yusuf	3	Neutral
8	mamat gratis	5	Positive
9	Nur Hidayati	3	Neutral

Figure 4. Sentiment Labeling Results Based on Ratings

Based on the labeling of 9,552 data points, the sentiment label distribution shows a significant class imbalance. Positive labels dominate with 7,235 data points (75.7%), followed by negative labels with 1,558 data points (16.3%), and neutral labels with 759 data points (8.0%). This condition indicates that most Info BMKG app users provide positive ratings. However, the dominance of the positive class may cause the classification model to be biased toward the majority class and reduce its ability to recognize minority classes. Therefore, in the modeling stage, the class imbalance problem was addressed by applying the class_weight='balanced' parameter to the Decision Tree and Random Forest algorithms. This technique assigns different weights to each class based on its proportion in the dataset, so minority classes receive higher attention during the training process. The use of this approach aims to improve the model's ability to classify all sentiment categories more fairly. The sentiment label distribution is shown in Figure 5.

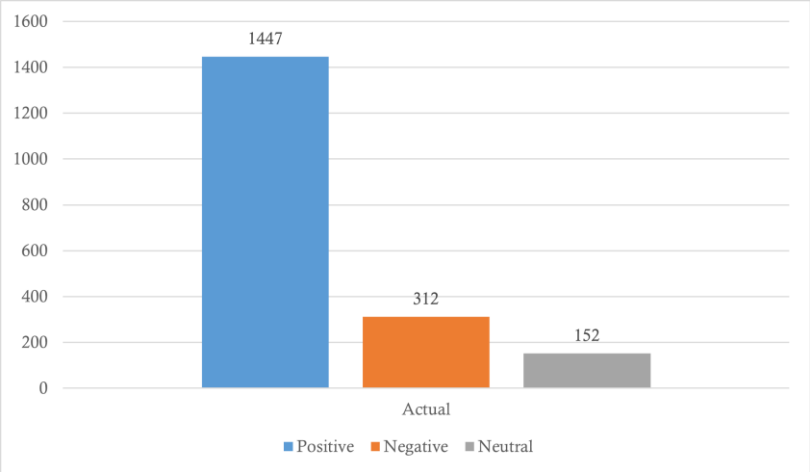

Figure 5. Distribution of Initial Sentiment Labels

3.1.4 Feature Extraction (TF-IDF)

The preprocessed text data must first be converted into a numerical format before it can be recognized by machine learning algorithms. The method used is TF-IDF. This process is performed using the TfidfVectorizer from the scikit-learn library with the parameter `max_features=5000`, so that only the 5,000 words with the highest weights are used as features.

The TF value is calculated based on the average frequency of each term across all documents, while the IDF value is obtained directly from the TfidfVectorizer after the data fitting process. The final TF-IDF value is the product of the TF and IDF values for each term. The weighting produces a matrix of size 9,552 × 5,000, meaning there are 9,552 review documents with 5,000 word features. The TF, IDF, and TF-IDF values for the first 10 terms are shown in Figure 6.

	Term	TF	IDF	TF-IDF
0	aa	0.000108744870	9.065998231602	0.000985880795
1	aamiin	0.000263397820	8.085168978590	0.002129615883
2	aangat	0.000095756183	9.471463339710	0.000906951173
3	abai	0.000021022346	9.471463339710	0.000199112375
4	abal	0.000229496022	8.778316159150	0.002014588639
5	abar	0.000053238637	9.471463339710	0.000504247800
6	abc	0.000059373197	9.471463339710	0.000562351056
7	abdate	0.000188193116	8.778316159150	0.001652018672
8	abdetan	0.000033190153	9.471463339710	0.000314359315
9	abgred	0.000044661654	9.471463339710	0.000423011223

Figure 6. TF, IDF, and TF-IDF Values

As shown in Figure 6, a high IDF score indicates that a word has a low frequency of occurrence across the entire corpus, thereby serving as a unique identifier. Conversely, a significant TF value indicates a high frequency of occurrence for a word within a specific document. The combination of these two factors yields a TF-IDF weight that represents the importance of a word within the context of the entire review corpus.

To provide a visual representation of the most dominant words based on TF-IDF weights, a word cloud was generated and is shown in Figure 7. Based on the word cloud, the words "good," "help," "weather," "app," "earthquake," "information," and "accurate" reflect the main topics discussed by users in their reviews of the Info BMKG app.

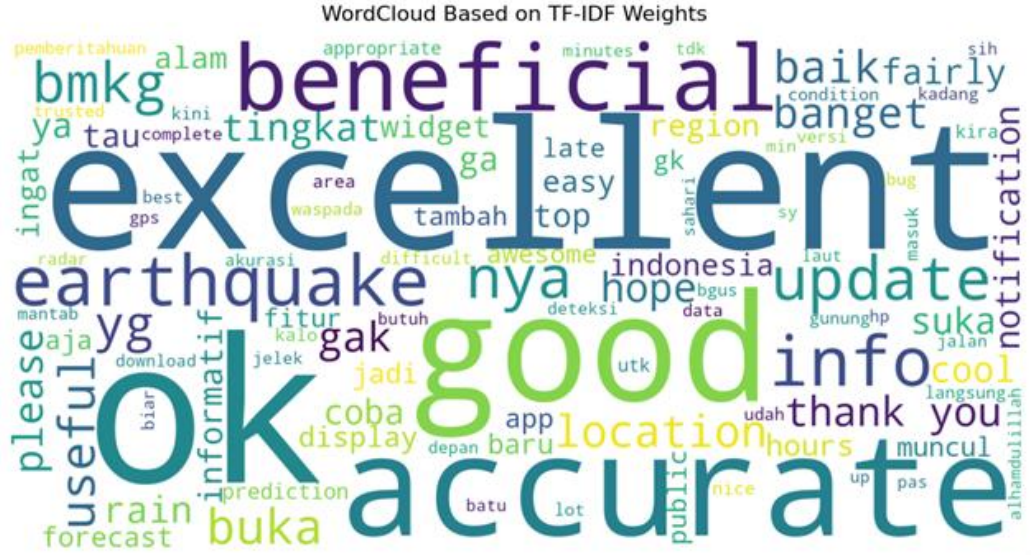

Figure 7. Wordcloud based on TF-IDF Weights

3.1.5 Data Splitting

The data was split using an 80:20 ratio, with 80% allocated for training and the remaining 20% for testing. The parameter `random_state=42` was applied to ensure consistency, while the parameter `stratify=y` was used to maintain class distribution balance. Details of the data distribution are presented in Table 1.

Table 1. Data Split

	Amount of Data	Amount of Features
Training	7.641	5.000
Testing	1.911	5.000

3.1.6 Modeling

The modeling phase was conducted to train and compare two classification algorithms, namely Decision Tree and Random Forest, using the TF-IDF feature matrix generated in the previous stage. The training data consisted of 7,641 review data points with 5,000 TF-IDF features, while the testing data consisted of 1,911 review data points. Both models were implemented using the scikit-learn library in the Jupyter Notebook environment.

In this stage, the modeling process was not only performed by applying the basic classifier configuration, but also by considering the imbalanced distribution of sentiment labels. Since the positive class dominated the dataset, while the negative and neutral classes had much smaller proportions, the class_weight='balanced' parameter was applied to the models. This parameter adjusts the weight of each class automatically based on its frequency in the training data, so that minority classes receive greater attention during the learning process. In addition, hyperparameter optimization was also conducted to obtain the best model configuration and improve classification performance.

3.1.6.1 Decision Tree

The Decision Tree model was built using the DecisionTreeClassifier algorithm. In the revised modeling process, the model was configured with class_weight='balanced' to reduce the effect of class imbalance on the learning process. Hyperparameter tuning was then applied to determine the most suitable parameter combination, such as the splitting criterion, maximum tree depth, minimum samples required for splitting, and minimum samples required at leaf nodes.

The Decision Tree algorithm works by recursively splitting the data based on the most informative features until a tree structure is formed. Each internal node represents a decision rule, while each leaf node represents the final sentiment class prediction. Although this algorithm has the advantage of being easy to interpret, it is still sensitive to data complexity and class imbalance. Therefore, the use of balanced class weights and parameter tuning was applied to improve its ability to recognize minority classes.

Based on the final modeling result, the Decision Tree model required 7.1717 seconds for the training process and 0.0991 seconds for the testing process. This shows that Decision Tree has relatively fast inference time because the prediction process only follows a single tree structure.

3.1.6.2 Random Forest

The Random Forest model was built using the RandomForestClassifier algorithm. Similar to Decision Tree, the revised Random Forest model also applied class_weight='balanced' to handle the imbalanced distribution of sentiment classes. Hyperparameter tuning was conducted to determine the optimal model configuration, including the number of trees, maximum depth, splitting criterion, and minimum samples for splitting and leaf nodes.

Random Forest works by constructing multiple decision trees and combining their predictions through a majority voting mechanism. This ensemble approach enables the model to produce more stable predictions compared to a single Decision Tree. In addition, the use of n_jobs=-1 allows the training process to run in parallel by utilizing all available processor cores, making the computation more efficient.

Based on the final modeling result, the Random Forest model required 2.7800 seconds for the training process and 0.1999 seconds for the testing process. Although its testing time was slightly longer than Decision Tree because predictions must be aggregated from multiple trees, Random Forest achieved a faster training time due to the parallelization process. A comparison of computational time between both models is presented in Table 2.

Table 2. Computing Time

	Training Time (s)	Testing Time (s)
Decision Tree	7.1717	0.0991
Random Forest	2.7800	0.1999

3.1.7 Evaluation Model

Model evaluation was conducted to assess the performance of the Decision Tree and Random Forest algorithms in classifying user review sentiment. The evaluation utilized a test dataset with several evaluation metrics and was supported by visualizations and ROC curves. The outputs of both models are shown in Figure 8.

	Model	Global Accuracy	Macro-Averaged Precision	Macro-Averaged Recall	Macro-Averaged F1-Score	CV Accuracy	Pure Training (s)	Testing Time (s)
0	Decision Tree	73%	51%	53%	52%	72%	7.1717	0.0991
1	Random Forest	78%	53%	55%	53%	79%	2.7800	0.1999

Figure 8. Comparison of Evaluation Metrics

The test results show that Random Forest outperforms Decision Tree across nearly all evaluation metrics. Random Forest achieved a global accuracy of 78%, macro-averaged precision of 53%, macro-averaged recall of 55%, and macro-averaged F1-score of 53%, while Decision Tree achieved a global accuracy of 73%, macro-averaged precision of 51%, macro-averaged recall of 53%, and macro-averaged F1-score of 52%. This indicates that Random Forest provides better overall classification performance, although the difference in macro-averaged F1-score between the two models is relatively small.

```
=== CLASSIFICATION REPORT: DECISION TREE ===
              precision    recall  f1-score   support

    negative       0.48      0.55      0.52       312
     neutral       0.17      0.20      0.19       152
    positive       0.88      0.84      0.86      1447

    accuracy                           0.74      1911
   macro avg       0.51      0.53      0.52      1911
weighted avg       0.76      0.74      0.75      1911


=== CLASSIFICATION REPORT: RANDOM FOREST ===
              precision    recall  f1-score   support

    negative       0.52      0.71      0.60       312
     neutral       0.21      0.09      0.13       152
    positive       0.90      0.88      0.89      1447

    accuracy                           0.79      1911
   macro avg       0.54      0.56      0.54      1911
weighted avg       0.78      0.79      0.78      1911
```

Figure 9. Classification Report of Decision Tree and Random Forest

The classification report provides a more detailed evaluation of model performance for each sentiment class. As shown in Figure 9, Random Forest achieved better overall performance than Decision Tree, particularly in the negative and positive classes. However, both models still showed weak performance in the neutral class, indicating that neutral sentiment remains difficult to classify due to its smaller data proportion and more ambiguous review patterns.

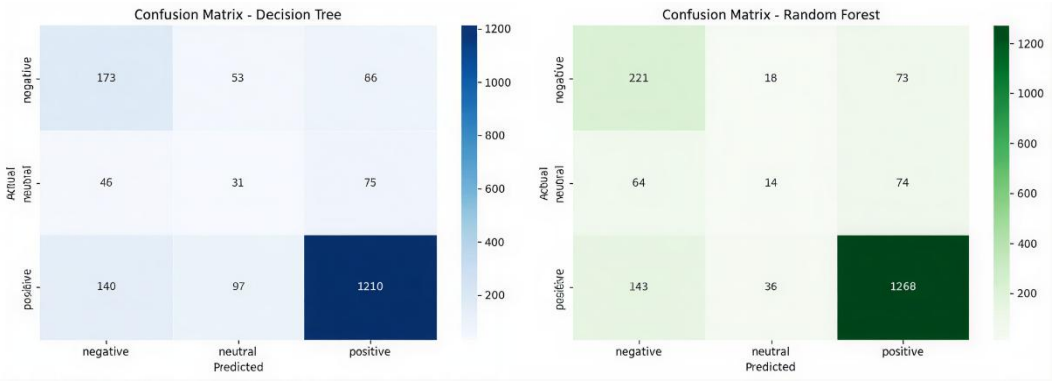

Figure 10. Confusion Matrix Comparison

As shown in the confusion matrix in Figure 10, Random Forest demonstrates better classification performance than Decision Tree in most sentiment classes. In the negative class, Decision Tree correctly classified 173 out of 312 data points, while Random Forest correctly classified 221 out of 312 data points, indicating better recognition of negative sentiment. For the positive class, Random Forest also achieved higher correct predictions, with 1,268 out of 1,447 data points, compared to Decision Tree with 1,210 correct predictions. However, both models still experienced difficulty in classifying the neutral class. Decision Tree correctly classified 31 out of 152 neutral data points, while Random Forest only correctly classified 14 neutral data points. This indicates that neutral sentiment remains the most challenging class to predict, likely due to its smaller data proportion and more ambiguous linguistic characteristics. Overall, the confusion matrix supports the evaluation results showing that Random Forest provides better classification performance than Decision Tree, particularly for the negative and positive classes.

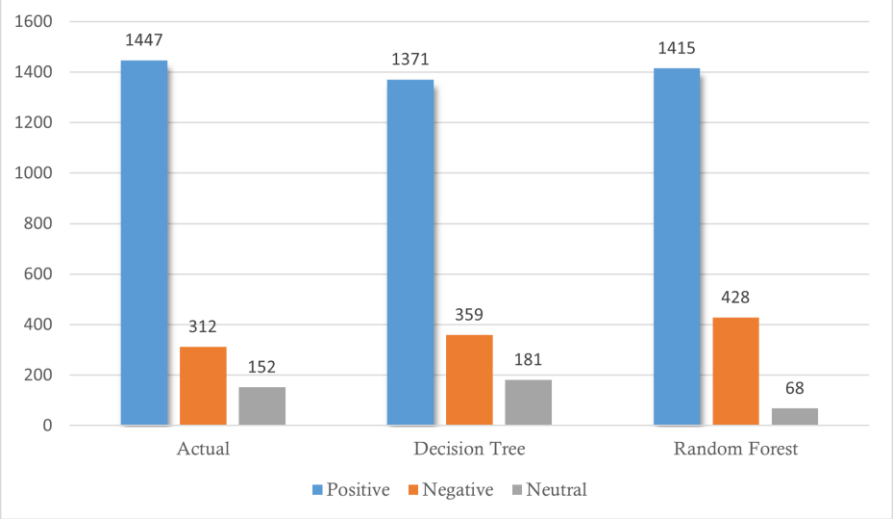

Figure 11. Comparison of Actual Values and Predicted Outputs

The validity of the prediction distribution is supported by Figure 11, which presents a comparison between actual values and predicted outputs. Random Forest produces a prediction distribution that is more consistent with the actual data than Decision Tree. This demonstrates Random Forest's superiority in maintaining class proportions in the dataset.

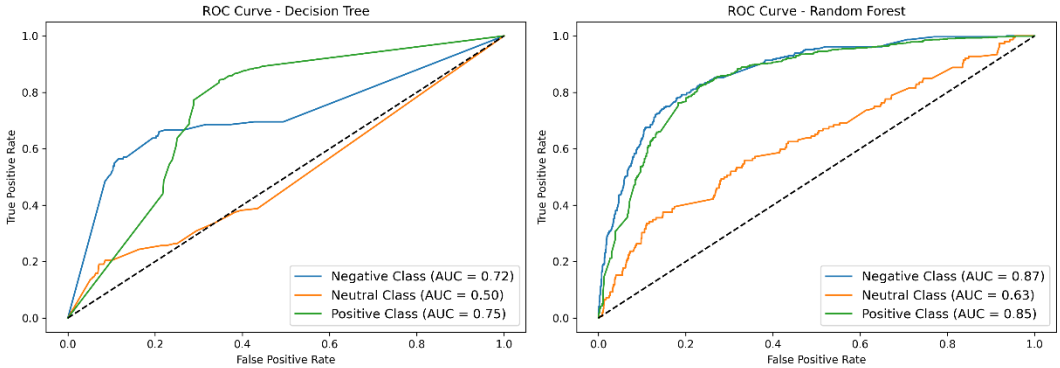

Figure 12. ROC Curve for Decision Tree and Random Forest

In terms of classification performance based on the ROC curve in Figure 12, Random Forest achieves higher AUC values than Decision Tree across all sentiment classes. In the negative class, Random Forest obtains an AUC of 0.87, while Decision Tree obtains 0.72. In the positive class, Random Forest also achieves a higher AUC of 0.85 compared to Decision Tree at 0.75. These results indicate that Random Forest has better discriminative ability in distinguishing negative and positive sentiment classes. However, the neutral class remains the weakest class for both models, with an AUC of 0.63 for Random Forest and 0.50 for Decision Tree. This low performance is likely influenced by the limited proportion of neutral data, which only accounts for 8.0% of the dataset, as well as the ambiguous nature of neutral reviews that often do not contain strong positive or negative sentiment expressions. As a result, the model has difficulty learning distinctive patterns for the neutral class, even after applying class balancing and hyperparameter tuning.

3.1.8 Analysis of Results

Based on the overall stages that have been carried out, this study produced several important findings regarding the data characteristics and the performance of the two classification models. The sentiment label distribution shows a strong dominance of the positive class, which accounts for 75.7% of the dataset, while the negative and neutral classes only account for 16.3% and 8.0%, respectively. This imbalance affects the learning process because the models tend to capture patterns from the majority class more easily than from minority classes.

After applying class balancing and hyperparameter tuning, Random Forest achieved better overall performance than Decision Tree. Random Forest obtained an accuracy of 78% and a macro-averaged F1-score of 53%, while Decision Tree achieved an accuracy of 73% and a macro-averaged F1-score of 52%. Although the difference in macro-averaged F1-score is relatively small, Random Forest shows stronger performance in the negative and positive classes, as reflected in the classification report and confusion matrix.

The confusion matrix shows that Random Forest correctly classified more negative and positive reviews than Decision Tree. In the negative class, Random Forest correctly classified 221 out of 312 data points, while Decision Tree correctly classified 173 data points. In the positive class, Random Forest correctly classified 1,268 out of 1,447 data points, compared to 1,210 correctly classified data points by Decision Tree. These results indicate that Random Forest has better capability in recognizing dominant and moderately represented sentiment patterns.

However, the neutral class remains the most difficult class for both models. Decision Tree correctly classified 31 out of 152 neutral data points, while Random Forest correctly classified only 14 neutral data points. This result is also supported by the ROC curve, where the neutral class produced the lowest AUC values, namely 0.50 for Decision Tree and 0.63 for Random Forest. The weak performance in this class is likely caused by the limited number of neutral samples and the ambiguous nature of neutral reviews, which often do not contain clear positive or negative expressions.

Overall, Random Forest provides better classification performance and stronger discriminative ability than Decision Tree, especially in the negative and positive classes. Meanwhile, Decision Tree

still offers an advantage in testing speed and model interpretability. These findings indicate that Random Forest is more suitable for sentiment classification of Info BMKG app reviews, although further improvement is still needed to improve the classification of neutral sentiment.

3.2 Discussion

This study was designed to compare the performance of Decision Tree and Random Forest in classifying user review sentiment for the Info BMKG app. The hypothesis formulated at the beginning of the study was that Random Forest would achieve better classification performance than Decision Tree because it uses an ensemble mechanism that combines multiple decision trees to produce more stable predictions.

The empirical results confirm this hypothesis. Random Forest achieved an accuracy of 78% and a macro-averaged F1-score of 53%, while Decision Tree achieved an accuracy of 73% and a macro-averaged F1-score of 52%. Although the difference in macro-averaged F1-score is relatively small, Random Forest showed stronger performance in the negative and positive classes. This indicates that the ensemble structure of Random Forest is more effective in capturing sentiment patterns from TF-IDF features than a single Decision Tree model.

These findings are consistent with previous studies showing that Random Forest tends to provide stable performance in sentiment classification tasks. Hapsari and Indriyanti [3] reported that Random Forest achieved strong performance in sentiment analysis of digital wallet applications, while Indriani and Wiranata [11] also showed that Random Forest outperformed Decision Tree and SVM in the sentiment analysis of GoPay application reviews. The consistency of these findings indicates that Random Forest is suitable for text classification tasks involving user reviews, especially in datasets with noisy and imbalanced characteristics.

The results also indicate that class imbalance handling and hyperparameter tuning can improve the fairness of model evaluation, especially when macro-averaged metrics are used. In this study, macro-averaged F1-score was included to avoid evaluation bias toward the majority class. Nevertheless, the performance of the neutral class remains limited, showing that balancing and tuning alone are not sufficient to fully solve the problem of minority class classification in sentiment analysis.

On the other hand, the Decision Tree model still shows limitations in handling complex and imbalanced data patterns. Although Decision Tree is easy to interpret and has faster prediction time, its structure as a single-tree model makes it more sensitive to data variation and minority class distribution. This finding is in line with Wulandari et al. [7], who showed that Decision Tree can achieve competitive performance in sentiment classification but still has potential weaknesses related to overfitting and instability when dealing with uneven data characteristics.

The results also confirm the importance of handling class imbalance in sentiment classification. The distribution of the Info BMKG review dataset is dominated by the positive class, while the negative and neutral classes have much smaller proportions. This condition can cause the model to focus more on the majority class and reduce its ability to recognize minority classes. Setiaji et al. [10] also emphasized that applying imbalance handling techniques can influence Random Forest performance and improve the model's ability to deal with uneven class distributions. Therefore, the use of class balancing and hyperparameter tuning in this study is relevant to produce a fairer evaluation of both models.

However, the classification results show that the neutral class remains the most difficult category to identify. This condition is reflected in the classification report, confusion matrix, and ROC curve, where the neutral class produced the lowest performance compared to the negative and positive classes. The weak performance of the neutral class may be caused by its limited data proportion, which only accounts for 8.0% of the dataset, as well as the linguistic ambiguity of neutral reviews. Neutral reviews often do not contain strong emotional expressions, making their textual patterns less distinct from positive or negative reviews.

From a practical perspective, Random Forest is recommended as the more suitable model for classifying Info BMKG app reviews because it provides better overall performance and stronger

discriminative ability in the negative and positive classes. Meanwhile, Decision Tree remains useful when model interpretability and faster prediction time are the main priorities. Therefore, the choice of model should consider both classification performance and computational efficiency.

For future research, several improvements can be explored. First, more advanced imbalance handling techniques such as SMOTE, ADASYN, or hybrid sampling methods can be applied to improve the representation of minority classes, especially the neutral class. Second, future studies may use deep learning approaches such as LSTM, BiLSTM, or transformer-based models like IndoBERT to capture semantic context more effectively than TF-IDF. Third, manual validation or hybrid labeling can be considered to reduce the limitations of rating-based sentiment labeling, particularly for reviews with neutral or ambiguous expressions. These directions are expected to improve classification performance and provide a more comprehensive sentiment analysis framework for public service applications.

CONCLUSION

This study demonstrates that the Random Forest algorithm performs better than Decision Tree in classifying sentiment from Info BMKG app reviews collected from the Google Play Store. Using 9,552 Indonesian-language review data points that had undergone preprocessing, rating-based sentiment labeling, TF-IDF feature extraction with 5,000 selected features, class imbalance handling, and hyperparameter optimization, Random Forest achieved an accuracy of 78% and a macro-averaged F1-score of 53%, while Decision Tree achieved an accuracy of 73% and a macro-averaged F1-score of 52%. The evaluation results also show that Random Forest has stronger performance in identifying negative and positive sentiment classes, supported by higher AUC values of 0.87 for the negative class and 0.85 for the positive class, compared to Decision Tree with AUC values of 0.72 and 0.75, respectively. However, both models still face limitations in classifying the neutral class, as reflected by the lower AUC values of 0.63 for Random Forest and 0.50 for Decision Tree, indicating that neutral reviews remain difficult to classify due to their limited data proportion and ambiguous linguistic patterns. Overall, Random Forest is recommended as the more suitable algorithm for sentiment classification of public service application reviews, while future research may explore more advanced imbalance handling techniques, deep learning models such as LSTM, BiLSTM, or IndoBERT, and richer text representation approaches to improve classification performance, especially for minority sentiment classes.

REFERENCES

[1] N. Aditiya, P. Setiaji, and Supriyono, "Analisis Sentimen Kepuasan Masyarakat terhadap Aplikasi "INFO BMKG" menggunakan Naive Bayes, SVM, dan KNN," *Sist. J. Sist. Inf.*, vol. 14, no. 03, pp. 1418–1432, 2025, doi: 10.32520/stmsi.v14i3.5223.

[2] A. A. Purnama and Y. R. Sipayung, "Sentiment Analysis of Public Service Using Naive Bayes Classifier," *J. Inf. Syst. Informatics*, vol. 7, no. 3, 2025, doi: 10.51519/journalisi.v7i3.1207.

[3] N. A. Hapsari and A. D. Indriyanti, "Analisis Sentimen pada Aplikasi Dompet Digital Menggunakan Algoritma Random Forest," *JEISBI (Journal Emerg. Inf. Syst. Bus. Intell.*, vol. 04, no. 03, pp. 186–192, 2023, doi: 10.26740/jeisbi.v4i3.55696.

[4] V. Fransisco and D. B. Rarasati, "Analisis Sentimen Aplikasi Polri Super App Menggunakan Algoritma Random Forest," *J. Ilm. Sains dan Teknol.*, vol. 8, no. 2, 2024, doi: 10.47080/saintek.v8i2.3383.

[5] A. Amelia, L. N. Hayati, and H. Darwis, "Analisis Sentimen Masyarakat Terhadap Sistem Pembayaran Mypertamina dengan Metode Random Forest , SVM , dan Naïve Bayes," *LINIER Lit. Inform. dan Komput.*, vol. 1, no. 1, pp. 28–44, 2024, doi: 10.33096/linier.v1i1.2269.

[6] M. H. Setiono, "Komparasi Algoritma Decision Tree, Random Forest, Svm Dan KNN dalam Klasifikasi Kepuasan Penumpang Maskapai Penerbangan," *INTI Nusa Mandiri*, vol. 17, no. 1, pp. 32–39, 2022, doi: 10.33480/inti.v17i1.3420.

[7] Wulandari, Nofiyani, and Y. P. Dewi, "Analisis Sentimen terhadap Ulasan Aplikasi Canva di Play Store dengan Menggunakan Pendekatan Lexicon dan Algoritma Decision Tree," *TICOM Technol. Inf. Commun.*, vol. 13, no. 02, pp. 57–63, 2025, doi: 10.70309/ticom.v13i2.133.

[8] U. Firdaus, A. Alfiah, and L. Mohdo, "Analisis Kinerja Decision Tree dan Random Forest Menggunakan Dataset Breast Cancer," *J. Pendidik. Sains dan Komput.*, vol. 6, no. 01, pp. 37–42, 2026, doi: 10.47709/jpsk.v6i01.7892.

[9] D. N. I. Huda, C. Prianto, and R. M. Awangga, "Analisis Sentimen Perbandingan Layanan Jasa Pengiriman Kurir Pada Ulasan Play Store Menggunakan Metode Random Forest dan Decision Tree," *J. Ilm. Inform.*, vol. 11, no. 02, 2023, doi: 10.33884/jif.v11i02.7952.

[10] G. G. Setiaji, J. Suntoro, and A. Rifa'i, "Random State Parameter Undersampling untuk Penanganan Data dengan Kelas Tidak Seimbang pada Algoritme Random Forest," *Transform. (Jurnal Transform.*, vol. 21, no. 2, pp. 73–83, 2024, doi: 10.26623/transformatika.v21i2.8901.

[11] Indriani and A. D. Wiranata, "Comparison of Accuracy Levels of SVM, Decision Tree and Random Forest Algorithms in Sentiment Analysis of User Responses of The Gopay Application," *J. Tek. Inform.*, vol. 5, no. 3, pp. 777–787, 2024, doi: 10.52436/1.jutif.2024.5.3.1885.

[12] M. F. Sodiq and F. Amin, "Sentiment Analysis of BMKG Weather Information Service Using K-Nearest Neighbor Method," *Int. J. Softw. Eng. Comput. Sci.*, vol. 4, no. 2, pp. 822–835, 2024, doi: 10.35870/ijsecs.v4i2.2881.

[13] R. Rahmadani, A. Rahim, and Rudiman, "Analisis Sentimen Ulasan 'Ojol The Game' di Google Play Store Menggunakan Algoritma Naïve Bayes dan Model Ekstraksi Fitur TF-IDF untuk Meningkatkan Kualitas Game.," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 12, no. 3, pp. 2928–2936, 2025, doi: 10.23960/jitet.v12i3.4988.

[14] S. J. Angelina, A. Bijaksana, P. Negara, and H. Muhardi, "Analisis Pengaruh Penerapan Stopword Removal Pada Performa Klasifikasi Sentimen Tweet Bahasa Indonesia," *JUARA (Jurnal Apl. dan Ris. Inform.*, vol. 02, no. 1, pp. 165–173, 2023, doi: 10.26418/juara.v2i1.69680.

[15] M. F. Y. Herjanto and Carudin, "Analisis Sentimen Ulasan Pengguna Aplikasi SIREKAP pada Play Store Menggunakan Algoritma Random Forest Classifier," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 12, no. 2, pp. 1204–1210, 2024, doi: 10.23960/jitet.v12i2.4192.

[16] M. H. Mahendra, D. T. Murdiansyah, and K. M. Lhaksmana, "Analisis Sentimen Tweet COVID-19 Menggunakan Metode K-Nearest Neighbors dengan Ekstraksi Fitur TF-IDF dan CountVectorizer," *Dike J. Ilmu Multidisiplin*, vol. 1, no. 2, pp. 37–43, 2023, doi: 10.69688/dike.v1i2.35.

[17] I. P. W. K. Gumi and A. Syafrianto, "Perbandingan Algoritma Naïve Bayes dan Decision Tree Pada Sentimen Analisis," *IJCSR (The Indones. J. Comput. Sci. Res.*, vol. 1, no. 2, pp. 1–15, 2022, doi: 10.59095/ijcsr.v1i2.11.

[18] I. Zulfahmi, "Analisis Sentimen Aplikasi PLN Mobile Menggunakan Metode Decission Tree," *J. Penelit. Rumpun Ilmu Tek.*, vol. 3, no. 1, pp. 11–21, 2024, doi: 10.55606/juprit.v3i1.3096.