INTERNATIONAL JOURNAL OF SOCIAL SCIENCE, EDUCATIONAL, ECONOMICS, AGRICULTURE RESEARCH AND TECHNOLOGY (IJSET) | ISSN (E): 2827-766X
Website:https://ijset.org/| Volumes 5 No. 3 (2026)

IMPLEMENTATION OF GEMINI PRE-PROCESSING ON 2024 SIREKAP REVIEWS USING THE RANDOM FOREST ALGORITHM

Amru Omar[1]*, Naufal Azmi Verdikha[2], Muhamad Ridwan[3]
Program Studi Teknik Informatika, Fakultas Sains Dan Teknologi,
Universitas Muhammadiyah Kalimantan Timur[1,2,3]
E-mail: 2211102441037@umkt.ac.id[1]*, nav651@umkt.ac.id[2], mr624@umkt.ac.id[3]

Received:01 /01/2026 | **Revised**: 20/01/2026 | **Accepted:** 10 /02/2026 | **Published**: 28/02/2026

Abstract

This study aims to classify reviews of the SIREKAP 2024 application by utilizing Large Language Model (LLM)-based Gemini pre-processing, Term Frequency–Inverse Document Frequency (TF-IDF) feature extraction, and the Random Forest algorithm as the classification method. The data used consist of user reviews obtained from the Google Play Store and categorized into five rating classes. Model performance evaluation was conducted using the 10-Fold Cross-Validation method with the Macro F1-Score metric. The testing results indicate that the lowest F1-Score achieved was 31.87%, while the highest reached 37.28%, with an overall average Macro F1-Score of 34.62%. These findings demonstrate that the Random Forest algorithm is capable of producing relatively stable classification performance through its ensemble learning mechanism, which combines multiple decision trees. However, its performance is still influenced by the imbalance in data distribution across classes. Therefore, Random Forest plays a role in maintaining prediction stability and reducing overfitting, although further development is required to improve classification performance on imbalanced review data.

Keywords: Gemini, Random Forest, Review Classification, SIREKAP, Cross-Validation, Data Imbalance

INTRODUCTION

SIREKAP is an electronic application developed to accelerate the real-time vote recapitulation process in Indonesia (Lestari et al., 2024). Although its objective is well-intentioned, in practice the application still faces various technical issues, such as delays in data processing, which have triggered numerous negative user reviews (Azzahri, 2024). In addition, users have reported implementation challenges, limited internet connectivity, and high server loads (Putri et al., 2025). To gain a deeper understanding of user-reported issues, review classification models are widely used to group comments based on specific themes or sentiments (Hanafi & R, 2024). Through such classification, developers can more quickly identify the main problems experienced by users in order to improve the SIREKAP application (Muthmainnah, 2025). A pre-processing stage is necessary because unstructured data can affect classification results; therefore, systematic steps such as text cleaning, normalization, and tokenization are required (Devia & Jariah, 2023).

Gemini is an AI-based model designed to enhance text representation quality in classification and prediction tasks. This model is considered more adaptive in performing data cleaning, normalization, and text tokenization. Apriliah et al. (2021) found that the utilization of Gemini improved the accuracy of text-based models across various digital platforms, making its application relevant for user review analysis. Zhang et al. (2023) also highlighted the use of Large Language Models (LLMs) in the data pre-processing stage. Their study showed that LLMs can clean text, correct writing errors, and adjust data context automatically without rigid rules. This approach transforms pre-processing from merely a technical step into one that can understand meaning and relationships between words in a text. Meguellati et al. (2025) further demonstrated that LLMs can improve data quality prior to analysis by recognizing and correcting contextual errors more naturally than traditional methods. Based on these findings, Gemini can be seen as an LLM implementation capable of supporting pre-processing, particularly in handling textual reviews such as those of the SIREKAP 2024 application. In line with this, Ridhoi et al. (2025) showed that the transformer-based model DistilBERT applied to SIREKAP reviews achieved an average F1-Score of 36.62%, with a highest performance of 37.16%. The F1-Score reflects the balance between precision and recall and is more informative than accuracy alone when dealing with imbalanced data.

Saputra and Saragih (2022) discussed application rating classification on Google Play Store, focusing on grouping app ratings based on characteristics and attributes such as the number of reviews and ratings. Meanwhile, A. Aziz and Zakir (2022) demonstrated that AI technology in review data processing can accelerate analysis while improving classification accuracy. Additionally, W. A. Aziz (2021) proved that the Random Forest algorithm can classify application reviews effectively, achieving an accuracy of 88.4%, with evaluation results from the confusion matrix and AUC-ROC curve indicating stable and consistent performance. Random Forest is an ensemble-based algorithm that constructs multiple decision trees and combines their predictions through a voting mechanism. Its advantages include reducing the risk of overfitting, maintaining stability with both large and small datasets, and providing accurate classification results without complex parameter tuning. In this study, the application of Random Forest to the pre-processing and classification of SIREKAP 2024 data is considered reliable, as supported by Prakoso Indaryono (2024), whose rainfall classification study in Indonesia achieved an accuracy of 86.55% and an F1-score of 86.5%. This suggests that Random Forest has strong potential to support election-related data analysis, particularly in ensuring accurate and stable classification results as a basis for evaluating the SIREKAP system.

Although previous research has applied DistilBERT, the relatively low F1-score indicates that review classification performance remains suboptimal. Therefore, this study focuses on implementing LLM-based Gemini pre-processing, TF-IDF feature extraction, and the Random Forest algorithm to classify SIREKAP 2024 application reviews. The review data were obtained from the study conducted by Ridhoi et al. (2025). This approach aims to examine the extent to which LLM support in the pre-processing stage can improve classification accuracy compared to manual pre-processing methods. Model performance is evaluated using the F1-Score metric to measure the balance between precision and recall in imbalanced review data. The research problem in this study is: What is the F1-score evaluation result obtained from the SIREKAP 2024 application rating review prediction model after undergoing pre-processing using Gemini 2.0 Flash, TF-IDF feature extraction, and the Random Forest algorithm. The objective of this study is to analyze the F1-score evaluation results of the SIREKAP 2024 application rating review prediction model using Gemini 2.0 Flash-based pre-processing, TF-IDF feature extraction, and the Random Forest algorithm.

LITERATURE REVIEW

Random Forest

Random Forest is a classification method that combines multiple decision trees trained in parallel, and their results are aggregated to produce the final prediction. This method applies the bagging technique, where each tree learns from a different subset of data, making the model more stable and less prone to overfitting. One of the main advantages of Random Forest is its ability to generate accurate predictions with more efficient training time compared to a single decision tree model (Shanmugasundar et al., 2021).

Hyperparameter Tuning with Optuna

Hyperparameter tuning is an important process in machine learning aimed at determining the best values for a model's external parameters. Unlike internal parameters that are automatically learned during training, hyperparameters must be defined before the training process begins. Selecting appropriate hyperparameters directly affects the model's ability to minimize errors, avoid overfitting or underfitting, and improve generalization to new data. In research conducted by Ridwan & Utami (2026), hyperparameter optimization was performed using Optuna, an adaptive hyperparameter optimization framework based on iterative evaluations (trials). Optuna explores the hyperparameter search space by evaluating each parameter combination according to the model's objective function value. The search process is dynamic, meaning that results from previous trials are used to determine the next hyperparameter combinations. The classification process using RandomForestClassifier begins by importing the required modules from the Scikit-learn library. The model is then initialized with specified parameters such as *n_estimators*, *max_depth*, and *class_weight*. The training process is performed using the training dataset to enable the model to learn the relationship between TF-IDF feature representations and their corresponding class labels. The trained model is subsequently used to predict labels for unseen test data. To measure performance, the model's predictions are evaluated using the F1-Score metric, which assesses the balance between precision and recall.

METHOD

The object of this study is user reviews of the SIREKAP 2024 application available on the Google Play Store platform. The review data were obtained from previous research conducted by Ridhoi et al. (2025) using a scraping method on February 6, 2024, at 10:00 PM WITA. A total of 8,358 reviews were collected, containing user names,

timestamps, comments, and ratings, with an average rating of 2.7 out of 5 stars. The dataset consists of the following attributes:

1. Name: Contains the name or username of the account that submitted the review.
2. Rating: Displays the user's rating on a scale of 1–5.
3. Time: Indicates when the review was created or posted.
4. Comment: Contains the textual content of the user's review.
5. Like: Shows the number of other users who found the review helpful.

The distribution of review data based on ratings is presented in Table 1, highlighting a significant class imbalance.

Table 1. Distribution of SIREKAP 2024 Reviews by Rating

Rating	Number of Reviews	Percentage
1	5,361	64.1%
2	736	8.8%
3	478	5.7%
4	255	3.1%
5	1,528	18.3%
Total	**8,358**	**100%**

This study utilized both hardware and software to support the research process. The hardware employed was one unit of an Asus TUF Gaming FX505e laptop equipped with an 8th Generation Intel Core i7 processor, 16 GB of RAM, and a 1 TB SSD internal storage. These specifications were considered adequate to handle data processing, model training, and evaluation efficiently. In addition, the software used in this study included Visual Studio Code as the primary code editor and various Python libraries to perform data processing, feature extraction, model development, and performance evaluation. Table 2 presents the list of Python libraries used in this study, including their respective versions and brief descriptions of their main functions.

Table 2. Python Libraries

Python Library	Version	Description
Pandas	1.4.4	Data manipulation and analysis
NumPy	1.21.5	Numerical computation and array operations
Matplotlib	1.21.5	Data visualization in graphical form
NLTK (Natural Language Toolkit)	2.8.1	Natural Language Processing (NLP)
Scikit-learn	1.0.2	Machine learning and classification algorithms
Google-generativeai	0.8.5	Access to Google's generative AI models

Table 2 details the software and hardware used in this research, as the availability of these tools is essential to support the research process. This study was conducted through structured stages to achieve the predefined objectives. The process began with data collection, followed by pre-processing using Gemini 2.0 Flash, feature extraction with TF-IDF, and classification using the Random Forest method. Finally, model performance evaluation was carried out. The sequence of these stages is illustrated in Figure 1.

Figure 1. Research Stages

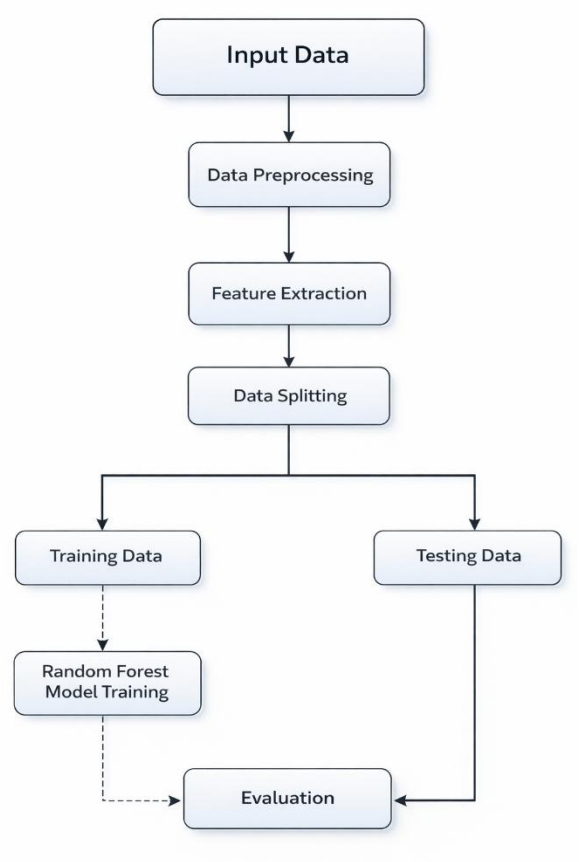

Figure 1 presents the research flow diagram. Initially, textual review data obtained from the Google Play Store were processed through pre-processing using the Gemini 2.0 Flash model. This process aims to remove irrelevant characters, perform tokenization, normalization, and prepare the data in a structured format suitable for further analysis in feature extraction and classification (Devia & Jariah, 2023). Subsequently, feature extraction was performed using TF-IDF to transform unstructured text data into numerical representations that can be processed by classification algorithms. The TF-IDF method was chosen because it assigns higher weights to words considered important within documents, as also demonstrated in the study by Ramadhansyah et al. (2024), which showed the effectiveness of TF-IDF in text-based sentiment analysis. Finally, the dataset was divided into training and testing data before training the Random Forest model and evaluating its performance.

RESULTS AND DISCUSSION

Random Forest Classification Results and Evaluation

Model performance evaluation was conducted using the K-Fold Cross Validation method with K = 10 folds. This method was applied to enhance the reliability of the testing results and to reduce potential bias that may arise from a single train–test split. The dataset consisted of 8,314 data entries, which were randomly divided into 10 folds by applying the parameters *shuffle=True* and *random_state=42*. These settings were intended to maintain consistency in the data splitting process and to ensure that the data distribution in each fold remained relatively balanced. Table 3.7 presents sample data indices used in fold-1 and fold-10.

Table 3.Train and Test Data Indices Results

Fold	Data	Data Indices	Total
1	Train Data	[0, 1, 2, 3, 4, …, 8310, 8311, 8313]	7,482
	Test Data	[23, 33, 37, 48, 61, ..., 8277, 8282, 8283, 8312]	832
2	Train Data	[0, 1, 2, 3, 4, ..., 8311, 8312, 8313]	7,482
	Test Data	[17, 19, 26, 31, 41, ..., 8275, 8280, 8284, 8291, 8307]	832
3	Train Data	[1, 2, 3, 4, 5, ..., 8311, 8312, 8313]	7,482
	Test Data	[0, 8, 14, 15, 34, ..., 8245, 8264, 8292, 8298, 8304]	832
4	Train Data	[0, 1, 2, 3, 4, ..., 8311, 8312, 8313]	7,482
	Test Data	[12, 29, 30, 47, 71, ..., 8234, 8237, 8255, 8281, 8290]	832
5	Train Data	[0, 1, 2, 3, 4, ..., 8311, 8312, 8313]	7,483
	Test Data	[39, 43, 49, 62, 68, ..., 8243, 8259, 8306, 8308, 8309]	831
6	Train Data	[0, 1, 2, 3, 4, ..., 8311, 8312, 8313]	7,483
	Test Data	[6, 18, 24, 25, 32, ..., 8286, 8289, 8293, 8294, 8305]	831
7	Train Data	[0, 1, 2, 3, 4, ..., 8309, 8311, 8312]	7,483
	Test Data	[7, 20, 22, 35, 42, ..., 8285, 8299, 8303, 8310, 8313]	831
8	Train Data	[0, 3, 4, 5, 6, ..., 8310, 8312, 8313]	7,483
	Test Data	[1, 2, 10, 11, 13, ..., 8279, 8288, 8295, 8301, 8311]	831
9	Train Data	[0, 1, 2, 4, 6, ..., 8311, 8312, 8313]	7,483
	Test Data	[3, 5, 9, 40, 54, ..., 8201, 8248, 8276, 8287, 8302]	831
10	Train Data	[0, 1, 2, 3, 5, ..., 8311, 8312, 8313]	7,483
	Test Data	[4, 16, 64, 98, 114, ..., 8272, 8274, 8296, 8297, 8300]	831

Based on Table 3, in each testing iteration, the system automatically divided the dataset into two groups: training data and testing data. In each fold, approximately 7,482 samples were used as training data, while 832 samples were used as testing data (with slight variations of 831–832 samples due to uneven division). This process was carried out iteratively until all data had an equal opportunity to serve as testing data. The data indices used in the training and testing processes for Fold-1 illustrate how the mechanism operates. The Train Index represents the data used to train the Random Forest model, while the Test Index represents the data used to evaluate model performance in that specific fold. This division strictly follows the K-Fold Cross Validation mechanism, ensuring that the test data in Fold-1 were not involved in the model training process. Subsequently, the model training process was carried out using the Random Forest algorithm. To further improve model performance, this study performed hyperparameter optimization using Optuna. The search for the best parameter combination was conducted through 200 trials, resulting in an optimal Random Forest configuration for classifying application ratings based on the F1-Score. A summary of the best parameter search results is presented in Table 4.

Table 4. Results of Parameter Optimization Using Optuna

Trial	n_estimators	max_depth	min_samples_split	min_samples_leaf	F1-Score
0	465	46	18	7	0.3389
1	490	44	4	7	0.3009
2	474	19	17	5	0.3179
3	414	15	18	7	0.3155
4	428	18	12	5	0.3011
…	…	…	…	…	…
195	494	50	6	6	0.3444
196	494	50	6	6	0.3444
197	493	50	6	6	0.3428
198	445	50	6	6	0.3416
199	489	50	6	6	0.3426

Based on the experimental results shown in Table 3.9, the final parameters applied and their explanations are as follows:

- **n_estimators**: Conceptually, the larger the value of *n_estimators*, the more decision trees are involved in the ensemble voting process, making the model more stable and reducing prediction variance. Across all trials, the value of *n_estimators* ranged between 400–500, indicating that a relatively large number of trees is

required to handle high-dimensional TF-IDF-based text data. However, increasing the number of trees does not always lead to proportional performance improvement. This can be observed from several trials with nearly identical numbers of trees but different F1-Score values.

- **max_depth**: This parameter allows the model to learn more complex and specific patterns in textual data. Trials with higher *max_depth* values (such as 40–50) tended to achieve better F1-Scores compared to trials with lower values (e.g., 15). Conversely, overly shallow trees limit the model's ability to capture complex relationships among features, leading to underfitting.
- **min_samples_split**: A larger *min_samples_split* value makes node splitting more restrictive, resulting in simpler trees and increasing the risk of underfitting. This is reflected in trials with higher values (such as 9 or 10), which produced lower F1-Scores. On the other hand, smaller values provide greater flexibility for the model to learn data structures, although excessively small values may increase the risk of overfitting.
- **min_samples_leaf**: This parameter plays an important role in controlling overfitting. A very small *min_samples_leaf* value, such as 1, allows leaf nodes to contain only a single sample, making the model highly sensitive to noise in the data. This condition can reduce performance. In contrast, moderate values (such as 3–5) demonstrate a more optimal balance between flexibility and model stability.

In addition to analyzing the selected parameters, the model optimization process was also evaluated through graphical visualizations generated by Optuna to ensure the effectiveness of the hyperparameter search process.

Figure 2. Hyperparameter Optimization Results of the Random Forest Model Using Optuna

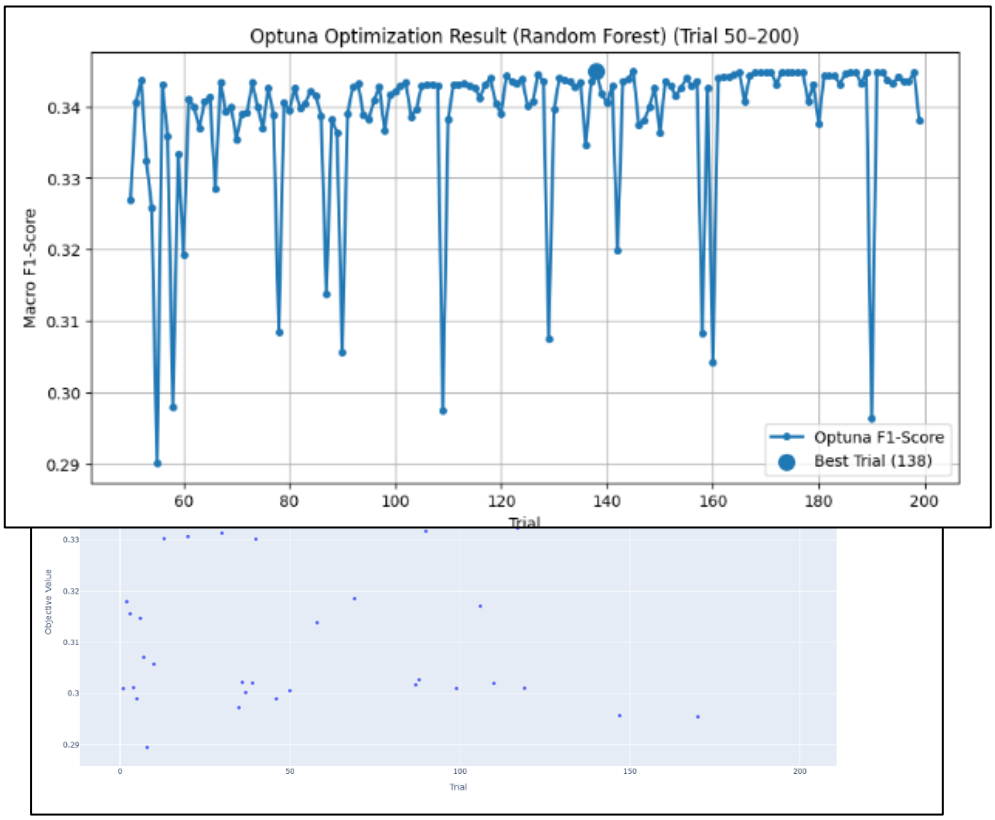

Figure 2 presents the visualization of hyperparameter optimization results produced by Optuna. The graph displays the Macro F1-Score values from trial 50 to trial 200 out of the total 200 trials conducted on the Random Forest model. Each point in the graph represents the Macro F1-Score achieved in a particular trial, reflecting the impact of different hyperparameter combinations on model performance. Although several trials show noticeable decreases in performance, overall the Macro F1-Score values tend to remain within a relatively consistent range. The point marked as the **Best Trial** indicates the optimal hyperparameter configuration, where the model achieved an F1-Score of 0.3462 at trial number 138. This result demonstrates that the hyperparameter combination in that trial provided an optimal balance between model complexity and generalization capability. After reaching this value, subsequent trials did not produce a significant improvement in performance.

Figure 3. Optimization History of the Random Forest Model

In Figure 3, it can be observed that the objective value in the form of the F1-Score increased during several early trials as Optuna explored different hyperparameter combinations. This improvement indicates that the optimization process successfully identified better parameter configurations compared to the initial settings. After reaching the best F1-Score of approximately 0.34, the best-value curve tended to stabilize until the final trial, suggesting that the optimization process had reached a convergence point and that adding more trials did not result in significant performance improvements.

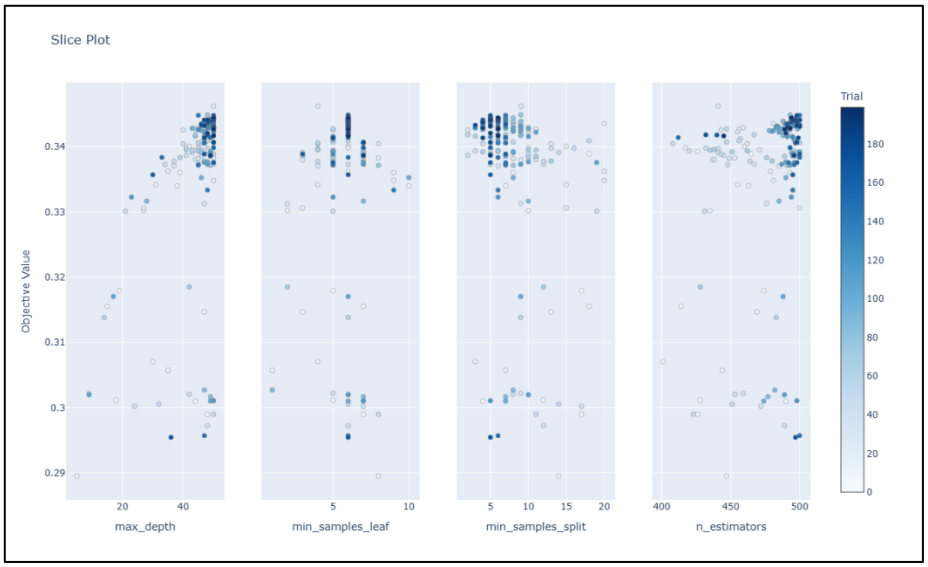

Figure 4. Slice Plot of Random Forest Parameters

Based on the Slice Plot shown in Figure 4, the influence of each hyperparameter on the resulting F1-Score can be analyzed. The *max_depth* parameter indicates that higher F1-Score values tend to be achieved at greater tree depths, suggesting that the Random Forest model requires deeper tree structures to capture complex patterns in textual data. The *min_samples_leaf* parameter shows an optimal range at low to moderate values, reflecting a balance between model generalization capability and overfitting prevention. Meanwhile, the *min_samples_split* parameter demonstrates a tendency toward optimal values in the low to moderate range, supporting flexibility in node splitting. Additionally, increasing the number of *n_estimators* tends to produce more stable F1-Score values, although performance improvements become limited beyond a certain threshold. The next step involved performing the Random Forest classification process, which is an ensemble learning method that combines multiple decision trees to generate final predictions. In this study, the Random Forest model was configured based on the optimal hyperparameters obtained from the tuning process, namely *n_estimators = 441*, *max_depth = 50*, *min_samples_split = 9*, and *min_samples_leaf = 4*, to address class imbalance across rating categories.

After completing the training and prediction processes across all 10 iterations, the next step was to calculate the model performance metrics using the F1-Score for each fold. The evaluation results are presented in Table 5.

Table 5. F1-Score Results for Each Fold

Fold	*F1-Score*
1	0,3728
2	0,3401
3	0,3513
4	0,3349
5	0,3287
6	0,3187
7	0,3501
8	0,3522
9	0,3484
10	0,3653
Rata-rata	0,3462

The table above presents the F1-Score evaluation results for each fold using 10-Fold Cross Validation. Based on these results, Fold 1 achieved the highest F1-Score of 0.3728, while the lowest score was obtained in Fold 6 with a value of 0.3187. The variation in F1-Score values across folds indicates differences in model performance depending on the data partition used in each iteration. To provide a visual representation of model performance stability, the comparison of F1-Score values across each fold is illustrated in Figure 5. The graph displays performance fluctuations for each testing fold, along with an average line representing the overall generalization capability of the Random Forest model, with an average F1-Score of 34.62%.

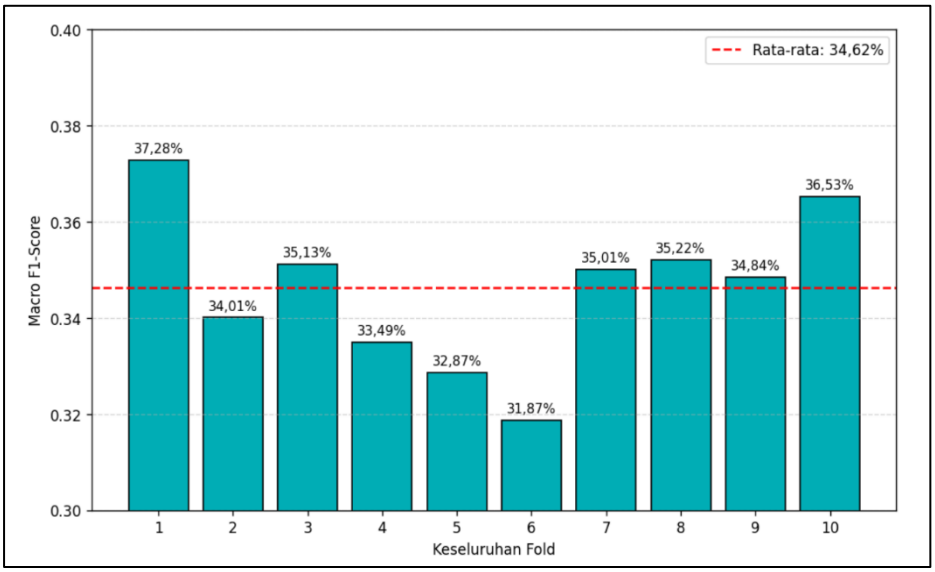

Figure 5. Average F1-Score Visualization

The F1-Score values ranging from 0.3728 to 0.3653 indicate that the performance of the Random Forest model remains relatively low. The average F1-Score of only 34.62% suggests that the model has not yet achieved a balanced classification performance. This is despite the use of relatively strict parameter configurations, such as a large number of trees and the implementation of balanced class weights. This condition indicates a high number of misclassifications in the form of false positives and false negatives, particularly in certain classes. Furthermore, the variation in F1-Score values reflects that the model's performance is not entirely stable, suggesting that the performance limitations are more influenced by the characteristics and quality of the data rather than by the model configuration itself.

To further evaluate classification errors in greater detail across the entire dataset, the prediction results from all 10 folds were combined into an Overall Confusion Matrix. This matrix maps the comparison between the actual labels and the predicted labels generated by the Random Forest model, providing deeper insight into which classes were classified accurately and which classes frequently experienced prediction errors. The results are shown in Figure 3.18.

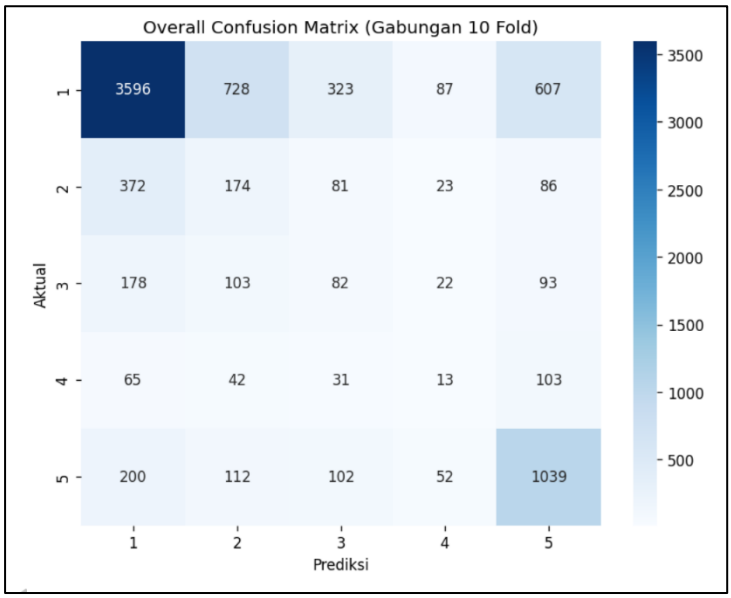

Figure 6. Overall Confusion Matrix (10-Fold)

Based on Figure 3.18, the Overall Confusion Matrix shows that the highest number of correct predictions (True Positives) occurs in Class 1, with 3,596 correctly classified instances, indicating the model's strong ability to recognize the majority class. In contrast, the lowest number of correct predictions appears in Class 4, with only 13 correctly classified instances, demonstrating the model's difficulty in classifying minority classes. Additionally, the matrix reveals significant misclassification patterns, where most data from minority classes, such as Classes 2, 3, and 4, tend to be incorrectly predicted as Class 1. This pattern indicates a bias toward the dominant class. This occurs even though the model employed optimally tuned hyperparameters obtained through the parameter search process. Therefore, it can be concluded that the primary issue lies not in the model configuration but in the imbalance of class distribution and the limited representation of minority class data. Overall, these results indicate that although hyperparameter optimization has been performed to improve the performance of the Random Forest model, the model performs well on classes with large amounts of data but still faces challenges in accurately classifying classes with fewer samples, resulting in imbalanced performance across classes.

Discussion

The initial stage of the study began with an analysis of the characteristics of the SIREKAP 2024 application review dataset. The results revealed a significant class imbalance, where reviews with a 1-star rating dominated the dataset at 64.1%, while 4-star ratings accounted for only about 3.1%. This condition presents a challenge in the classification process because the model tends to learn patterns from the majority class more easily and may overlook classes with fewer samples. Therefore, this study did not rely solely on accuracy as an evaluation metric but instead used the Macro F1-Score as the primary performance measure to ensure that the model's performance across all rating classes was assessed proportionally. The data pre-processing stage utilized the LLM Gemini 2.0 Flash model as the main tool for text processing. The model was employed to clean and normalize review data through several steps, including lowercasing, removing unnecessary characters, spell-checking, and stemming. The use of LLM at this stage aimed to improve text quality by correcting spelling errors and standardizing informal language variations commonly found in user reviews. After these steps, the review texts became more consistent, concise, and structured, thereby minimizing noise that could interfere with feature extraction and classification processes. The pre-processed text data were then transformed into numerical form using the Term Frequency–Inverse Document Frequency (TF-IDF) method. This method assigns weights to words based on their frequency within a document relative to the entire corpus. TF-IDF is effective in highlighting important and relevant words that reflect user evaluations of the SIREKAP application. However, this method has limitations because it relies solely on statistical word frequency

calculations without considering contextual meaning or relationships between words. This limitation affects the model's ability to understand reviews written in informal language, containing ambiguity, or expressing implicit meanings within ratings. In the classification stage, the Random Forest algorithm was used as the primary model due to its ability to combine multiple decision trees through an ensemble learning approach. This model was selected because it is relatively robust against overfitting and capable of handling high-dimensional data generated from TF-IDF feature extraction. To achieve optimal performance, hyperparameter optimization was conducted using the Optuna framework by exploring various parameter combinations, such as the number of trees and maximum tree depth. This process aimed to identify a model configuration that balances complexity and generalization capability on unseen data. Model performance evaluation was carried out using 10-Fold Cross-Validation to ensure objective and consistent results. The dataset was divided into ten subsets, with each subset used alternately as testing data while the remaining subsets served as training data. The evaluation results showed that the F1-Score for each fold ranged from 31.87% to 37.28%, with an average Macro F1-Score of 34.62%. These values indicate that the Random Forest model produced relatively stable performance, although its accuracy cannot yet be considered optimal.

The moderate F1-Score results were influenced by several key factors. First, the imbalance in rating class distribution caused the model to recognize majority classes more effectively than minority classes, despite the implementation of class weighting. Second, the limitations of TF-IDF in capturing sentence-level contextual meaning restricted the model's comprehensive understanding of review content. Thus, although data quality was improved through LLM-based Gemini pre-processing, classification performance remained strongly influenced by dataset characteristics and the chosen feature representation approach. Overall, the findings demonstrate that the implementation of Gemini 2.0 Flash in the pre-processing stage improved the quality and consistency of textual data while simplifying what was previously a manual cleaning process. Meanwhile, the use of the Random Forest algorithm provided relatively stable classification performance and helped reduce overfitting risk through its ensemble learning mechanism. However, the classification results were still affected by data imbalance and the limitations of TF-IDF-based feature extraction. Therefore, this study opens opportunities for further development, such as employing embedding-based text representation methods or deep learning approaches to enhance the model's ability to understand the semantic context of user reviews more comprehensively.

CONCLUSION

This study implements Large Language Model (LLM)-based preprocessing using Gemini 2.0 Flash on user review data from the SIREKAP 2024 application. This approach effectively processes unstructured review texts containing spelling errors and morphological variations through systematic and consistent stages, including lowercase transformation, removal of unnecessary characters, spellchecking, and stemming. As a result, the text data become cleaner and more structured. The processed data are then transformed into numerical representations using the Term Frequency–Inverse Document Frequency (TF-IDF) method, followed by classification using the Random Forest algorithm. Model performance is evaluated using the 10-Fold Cross Validation method to ensure objective testing results and good generalization capability. Based on the evaluation results, the classification model achieved an average F1-Score of 34.41%. These findings indicate that although LLM-based preprocessing improves text data quality, the classification performance remains influenced by class distribution imbalance and the limitations of the TF-IDF feature extraction method in capturing more complex semantic context. Therefore, it can be concluded that the use of the Gemini LLM in the preprocessing stage contributes to optimizing contextual consistency and preparing the dataset with systematic format standardization. However, this approach does not directly guarantee improved classification performance, particularly when the main challenge lies in the significant class distribution imbalance, where rating class 1 dominates more than 60% of the data. This condition causes the Random Forest model to exhibit bias toward the majority class, even though evaluation is conducted using the Macro F1-Score metric, which fairly considers all classes.

REFERENCES

Amelia Yoga Lestari, & Joy Nashar Utamajaya. (2024). Audit Sistem Informasi Aplikasi Sirekap KPU: Analisis Keamanan dan Efisiensi. *Switch : Jurnal Sains Dan Teknologi Informasi*, *2*(4), 23–32. https://doi.org/10.62951/switch.v2i4.178

Apriliah, W., Kurniawan, I., Baydhowi, M., & Haryati, T. (2021). Prediksi Kemungkinan Diabetes pada Tahap Awal Menggunakan Algoritma Klasifikasi Random Forest. *Sistemasi*, *10*(1), 163. https://doi.org/10.32520/stmsi.v10i1.1129

Aziz, A., & Zakir, S. (2022). *Indonesian Research Journal on Education: Jurnal Ilmu Pendidikan*. *2*(3), 1030–1037.

Aziz, W. A. (2021). Implementasi metode random forest pada klasifikasi data ulasan konsumen perusahaan (studi kasus: Aplikasi kai access). In *Repository.Uinjkt.Ac.Id*.

Azzahri, R. (2024). Tinjauan Kritis terhadap Penggunaan Aplikasi Sirekap dalam Proses Pemilihan Umum Presiden Tahun 2024. *Iapa Proceedings Conference*, 398. https://doi.org/10.30589/proceedings.2024.1067

Devia, E., & Jariah, A. (2023). Analisis Sentimen Review Aplikasi Video Conference Menggunakan Algoritma Support Vector Machine (Studi Kasus: Skype Dan Zoom). *Jurnal Information System*, *3*(2), 65–72.

Hanafi, M. R., & R, R. K. (2024). *Sentiment Analysis on Sirekap App Reviews on Google Play Using Naive Bayes Algorithm Analisis Sentimen pada Ulasan Aplikasi Sirekap di Google Play Menggunakan Algoritma Naive Bayes*. *4*(October), 1578–1586.

Larasati, F. A., Ratnawati, D. E., & Hanggara, B. T. (2022). Sentiment Analysis of Dana Application Reviews Using the Random Forest Method. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *6*(9), 4305–4313.

Meguellati, E., Pratama, N., Sadiq, S., & Demartini, G. (2025). Are Large Language Models Good Data Preprocessors? *WWW Companion 2025 - Companion Proceedings of the ACM Web Conference 2025*, 2129–2132. https://doi.org/10.1145/3701716.3717568

Muthmainnah, D. (2025). *KLASIFIKASI ULASAN APLIKASI SIREKAP 2024 DENGAN EKSTRAKSI FITUR WORD2VEC DAN METODE SUPPORT VECTOR MACHINE (SVM)*. *9*(2), 3013–3019.

Prakoso Indaryono, N. A. (2024). Analisa Perbandingan Algoritma Random Forest Dan Naïve Bayes Untuk Klasifikasi Curah Hujan Berdasarkan Iklim Di Indonesia. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *9*(1), 158–167. https://doi.org/10.29100/jipi.v9i1.4421

Putri, J. A., Sari, N. Y., & Rahayuningtyas, F. D. (2025). *EFEKTIVITAS PENGGUNAAN APLIKASI SIREKAP (SISTEM INFORMASI REKAPITULASI) DALAM PEMILU 2024*. *2*(2), 351–360.

Ramadhansyah, D., Asrofiq, A., & Yunefri, Y. (2024). Analisis Sentimen Ulasan penumpang maskapai penerbangan di Indonesia… ZONAsi. *Jurnal Sistem Informasi*, *6*(2), 287–297.

Ridhoi, R., Azmi Verdikha, N., & Yulianto, F. (2025). Analisis Klasifikasi Ulasan Aplikasi Sirekap 2024 menggunakan Ekstraksi Fitur DistilBert Dan Metode Support Vector Machine. *Jurnal Ilmiah Informatika (JIF)*.

Ridwan, M., & Utami, E. (2026). *Optimized Hyperparameter Tuning for Improved Hate Speech Detection*. *5*(158), 525–534.

Saputra, A. C., & Saragih, A. S. (2022). KLASIFIKASI RATING APLIKASI ANDROID DI GOOGLE PLAY STORE MENGGUNAKAN ALGORITMA GRADIENT BOOST Agus Sehatman Saragih. *Oktober*, *6*(1), 18–29.

Shanmugasundar, G., Vanitha, M., Čep, R., Kumar, V., Kalita, K., & Ramachandran, M. (2021). A comparative study of linear, random forest and adaboost regressions for modeling non-traditional machining. *Processes*, *9*(11). https://doi.org/10.3390/pr9112015

Zhang, H., Dong, Y., Xiao, C., & Oyamada, M. (2023). *Large Language Models as Data Preprocessors*. 3–6.