

Sentiment Analysis Google Gemini Reviews Using K-Nearest Neighbor and Particle Swarm Optimization

Siraj Fatwa Sidqy*
Information Systems
Faculty of Technology and Design
Universitas Pembangunan Jaya
South Tangerang, Indonesia
siraj.fatwasidqy@student.upj.ac.id

Yunus Widjaya
Information Systems
Faculty of Technology and Design
Universitas Pembangunan Jaya
South Tangerang, Indonesia
yunus.widjaya@upj.ac.id

Abstract—Google Gemini, Google's generative AI assistant, receives a large volume of user reviews on the Google Play Store daily, making manual reading of user opinions impractical. This study analyzes the sentiment of Indonesian-language Google Gemini reviews using K-Nearest Neighbor (KNN) as the baseline classifier, then optimizes the k parameter through Particle Swarm Optimization (PSO) to examine its effect on classification performance. A total of 5,000 reviews were collected via web scraping using the Google Play Scraper library between 1 June and 1 September 2026, followed by preprocessing (duplicate removal, cleaning, case folding, normalization, tokenization, and stopword removal), leaving 2,858 labeled reviews, of which 2,000 were used for modeling. Negative, Neutral, and Positive labels were assigned automatically from each reviewer's star rating. Text was represented using CountVectorizer, producing 2,282 features, split into 1,600 training and 400 testing samples via stratified sampling. The baseline KNN model with $k=5$ achieved 79.50% accuracy, while PSO selected $k=4$ with a best cost of 0.188125, yielding a KNN-PSO accuracy of 80.75%, a 1.25 percentage-point increase. This improvement was accompanied by a rise in macro F1-score from 0.441 to 0.458 and weighted F1-score from 0.765 to 0.780, mainly driven by the Negative and Positive classes, while the Neutral class remained undetected by both models (F1=0). These findings indicate that PSO optimization yields a consistent but limited improvement and does not resolve the underlying difficulty of identifying the Neutral class caused by imbalanced class distribution.

Keywords—Sentiment Analysis; Google Gemini; K-Nearest Neighbor; Particle Swarm Optimization; CountVectorizer



© 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution ShareAlike (CC BY SA) license (<https://creativecommons.org/licenses/by-sa/4.0/>).

questions to complaints about technical problems, response speed, and the accuracy of the answers it provides. Those reviews contain valuable information about user perception, but their large volume and unstructured nature make manual reading impractical for developers who want to monitor user satisfaction continuously, as generally identified in a systematic review of app reviews in software engineering [2].

This condition creates a need for sentiment analysis, a branch of text mining that automatically classifies opinions into categories such as positive, neutral, or negative. Through sentiment analysis, developers can obtain an overall picture of user opinion trends without having to read every review one by one. Various machine learning algorithms have been used for this purpose, among them Naïve Bayes, Support Vector Machine, Random Forest, and K-Nearest Neighbor (KNN), each of which has different characteristics and levels of complexity in handling high-dimensional text data [3], [4], [5], [6].

KNN was selected in this study because its mechanism is simple and does not require any particular assumption about the distribution of the data, so it is relatively easy to apply to text data that has been represented numerically [3], [7]. Even so, the performance of KNN depends heavily on the choice of the parameter k , that is, the number of nearest neighbors considered during the classification process. A value of k that is too small makes the model vulnerable to noise, while a value of k that is too large may blur the boundaries between classes, especially on data with an imbalanced class distribution [8], [9].

To reduce the dependence on selecting k manually, this study applies Particle Swarm Optimization (PSO), a population-based optimization algorithm inspired by the movement of bird flocks searching for food sources, and it has also been used to optimize classifiers in sentiment analysis research [5], [10]. PSO works by distributing a number of particles that represent candidate solutions, then updating the position of each particle based on its individual best experience (personal best) and the best experience of the group (global best) until the population converges. In this study, PSO is applied specifically to search for the most suitable value of k for the KNN algorithm based on an objective function in the form of cross-validation accuracy, without involving feature selection or the weighting of any other attribute.

I. INTRODUCTION

The rapid growth of generative artificial intelligence technology led Google to release Gemini as a multimodal conversational assistant that can be accessed through Android devices [1]. Since it became widely available, the application has received thousands of reviews every month on the Google Play Store, ranging from appreciation of its ability to answer

The contribution of this study lies in applying PSO specifically to determine the value of k in the KNN algorithm for classifying the sentiment of Google Gemini reviews, together with a transparent analysis of the effect of that optimization on each sentiment class, including the class with a relatively small amount of data. Based on that description, this study aims to apply the KNN algorithm to classify the sentiment of Google Gemini user reviews, to optimize the value of the parameter k using PSO, and to compare the performance of both models based on accuracy, precision, recall, and F1-score for each sentiment class. The results are expected to give an objective picture of the effectiveness of PSO optimization on KNN for a three-class sentiment classification case with an imbalanced data distribution, and at the same time serve as a reference for developers in using user reviews as material for continuous product evaluation.

II. LITERATURE REVIEW

The combination of KNN and PSO, as well as similar text mining methods applied to application reviews, has been studied through a variety of approaches. Ramadhan et al. [3] applied plain KNN to reviews of the Maxim application without an optimization stage, so the value of k was determined through manual trial and error. Nufairi et al. [11] and Kurniawan et al. [12] respectively tested Support Vector Machine and a comparison of Naïve Bayes, Support Vector Machine, and Random Forest on reviews of the Threads and Ajaib applications, but neither involved KNN or metaheuristic optimization. Vamilina and Novita [5] applied PSO to optimize Support Vector Machine in an e-wallet sentiment case, and Nugraha and Hardjianto [10] applied the same combination to Bukalapak service reviews on Twitter, which shows that the metaheuristic approach in this domain has mostly been directed at classification algorithms other than KNN. Rahmawati et al. [13] applied Naïve Bayes to reviews of the m-BCA application without parameter optimization and without any comparison against distance-based algorithms such as KNN. In the context of generative artificial intelligence applications, Rosandi et al. [14] compared several machine learning algorithms to analyze the sentiment of Gemini AI reviews and reported the best performance on margin-based models such as Support Vector Machine, without discussing KNN or metaheuristic optimization. Meanwhile, Akbar [15] compared Support Vector Machine and Naïve Bayes on three generative AI applications at once, namely ChatGPT, Google Gemini, and Microsoft Copilot, adding class balancing techniques in the form of SMOTE and Random Oversampling to handle the dominance of positive sentiment, but did not involve KNN or PSO in the classification process.

Those studies generally show that Naïve Bayes and Support Vector Machine are more often used as the main model in sentiment cases on generative AI applications, while the use of KNN optimized purely by PSO in this domain is still rarely found. Based on that review, there is a research gap in applying metaheuristic optimization to the KNN parameter in depth on reviews of generative AI applications, considering that most previous studies optimized SVM or Naïve Bayes, or used PSO as a feature selection mechanism instead of a parameter search.

As a result, the pure effect of optimizing the value of k on KNN performance in the domain of Indonesian-language generative AI application reviews is still not widely known, particularly under the imbalanced sentiment class distribution commonly found in Play Store review data.

III. METHOD

This study uses a quantitative approach with an experimental method to evaluate the performance of the K-Nearest Neighbor (KNN) algorithm before and after optimizing the value of the parameter k using Particle Swarm Optimization (PSO). All research stages, from data collection to model evaluation, follow the implementation flow of the research notebook as shown in Fig. 1.

A. Research Stages

The research stages are arranged sequentially as shown in Fig. 1, starting from data collection, preprocessing, sentiment labeling, feature representation, the training and testing data split, training of the baseline KNN model, PSO optimization, retraining of KNN using the optimized value of k , and finally the evaluation and comparison of the performance of both models.

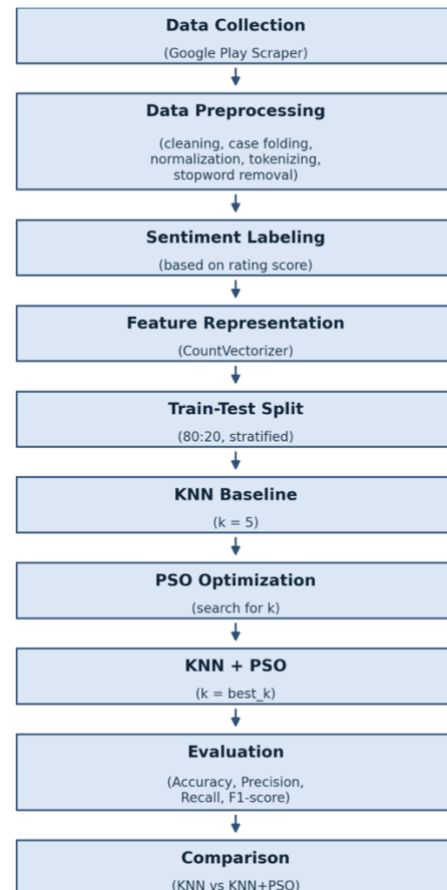


Fig. 1. Research Stages

B. Data Collection

Review data was collected through web scraping using the Python-based Google Play Scraper library, targeting the

application with the package name `com.google.android.apps.bard`, which is the official package identity of Google Gemini on the Google Play Store. Data retrieval was limited to Indonesian-language reviews (region id) uploaded by users between 1 June 2026 and 1 September 2026, with a maximum target of 5,000 most recent reviews. Each review record obtained consists of the username, the rating score (1–5), the review date, and the text of the opinion. The scraping process successfully collected 5,000 reviews in line with the target. It should be emphasized that those 5,000 reviews are the result of raw data collection and not the amount of data used directly for modeling, because the data still passed through several further filtering stages as explained in Sections III-C to III-F.

C. Text Data Preprocessing

Before being classified, the review text went through several preprocessing steps in a systematic order. First, duplicate data was removed based on identical content in the content column using the `drop_duplicates` function, considering that application users tend to write short repeated reviews such as “bagus” or “mantap”. Second, the cleaning stage removed links (URLs), user mentions (usernames), emoji, symbols, and numbers from the review text using regular expressions. Third, case folding converted all letters into lowercase so that the same word would not be treated as different because of capitalization. Fourth, the normalization of non-standard words was carried out by matching every word against a standard word dictionary downloaded from the `kamus_kata_baku` repository, in line with the dictionary-based approach also used in other research on the normalization of non-standard Indonesian words [16]. Fifth, tokenization split the text into word units based on spaces. Sixth, stopword removal deleted common Indonesian words using the stopword list from the NLTK library, plus a specific stopword list built by the researcher covering words such as “ya”, “ku”, “pas”, “gue”, “sih”, “bug”, “error”, and “app”. The removal of the words “bug” and “error” at this stage needs to be noted as a methodological decision, because both terms potentially carry relevant negative sentiment in application reviews, so they also affect the feature representation of the Negative class. No ablation experiment removing this stopword-list decision was conducted in this study, so its quantitative effect on classification performance is not empirically established here and is instead noted as a direction for further work in Section VI. Finally, as a last corrective pass applied to the stopword-removal result, a small additional dictionary built by the researcher replaced a few remaining specific terms, such as “apk” into “aplikasi” and “ngedit” into “edit”.

D. Sentiment Labeling

Sentiment labels were assigned automatically based on the rating score given by the user through the `label_sentimen` function, with the rule that scores 1 and 2 were labeled Negative, score 3 was labeled Neutral, and scores 4 and 5 were labeled Positive, as presented in TABLE 1. Rating-based labeling was chosen because it is efficient to apply to data consisting of thousands of reviews without requiring manual annotation, although this approach assumes that the numerical rating is always consistent with the content of the review text. No manual

annotation or sample-based validation of the automatically assigned labels (for example, using an agreement measure such as Cohen's Kappa) was carried out in this study, so the extent to which the rating-based labels align with the semantic sentiment expressed in the review text remains unverified and is treated here as a labeling-approach limitation rather than a validated ground truth.

TABLE 1. Sentiment Labeling Rules Based on Rating Scores

Score Range	Sentiment Label
1 – 2	Negative
3	Neutral
4 – 5	Positive

E. Feature Representation

The text representation was built using `CountVectorizer` from the Scikit-learn library, which converts each review resulting from stopword removal into a word frequency vector (bag-of-words) based on the vocabulary formed from the training data. Each component of the vector represents one unique word, with a value equal to the number of times that word appears in the document concerned. The vectorization process produced 2,282 features (vocabulary columns) that were used consistently to represent both the training and the testing data. The choice of `CountVectorizer`, instead of TF-IDF weighting, is consistent with the implementation in the research notebook and keeps the feature representation simple as a pure frequency count without any adjustment for word rarity across the whole corpus.

F. Training and Testing Data Split

From the 5,000 reviews obtained through scraping, the removal of duplicate data described in Section III-C left 2,959 unique reviews. After passing through all preprocessing stages, rows with empty (missing) values in any of the preprocessing columns were removed using `dropna`, leaving 2,858 labeled reviews ready to be used. Before splitting into training and testing data, this study took a subset of the first 2,000 rows out of those 2,858 labeled reviews as the modeling data, following the row order in which the data was already stored after preprocessing rather than through random sampling. Because the reviews were retrieved in `Sort.NEWEST` order during scraping (Section III-B) and this ordering was preserved through preprocessing, the first 2,000 rows correspond to the most recently posted reviews within the collection period; this chronological selection, and its implication for the representativeness of the modeling subset, is discussed further as a limitation in Section V. This subset was taken to keep the computational load under control, considering that the PSO optimization process requires repeated KNN training and cross-validation at every iteration and for every particle. The modeling data was then split into training and testing data with an 80:20 ratio using the `train_test_split` function, with the `stratify` parameter based on the sentiment label to keep the proportion of each class consistent between the training and testing data, and `random_state=42` so that the split can be reproduced. That split produced 1,600 training samples (80%) and 400 testing samples (20%), as summarized in TABLE 2 and visualized in Fig. 2

TABLE 2. Data Quantity at Each Processing Stage

Stage	Amount of Data
Scraping result (target 5,000)	5,000
After duplicate removal	2,959
After preprocessing and labeling (dropna)	2,858
Modeling data subset	2,000
Training data (80%)	1,600
Testing data (20%)	400

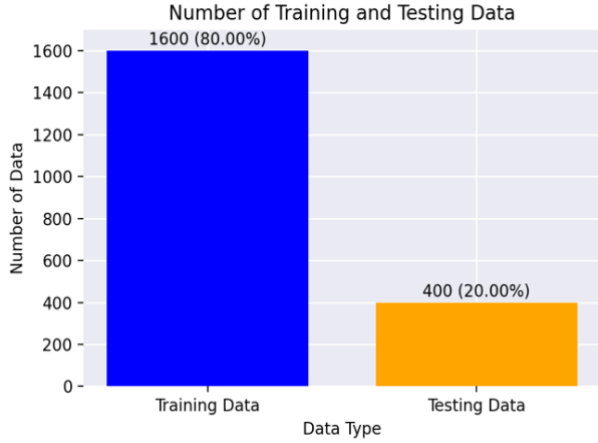


Fig. 2. Training and Testing Data Distribution

G. K-Nearest Neighbor Baseline

The baseline KNN model was built using the KNeighborsClassifier class with $k=5$ as an initial reference value commonly used in text classification research. Model training was performed using the vectorized training data, while testing was performed on the testing data to obtain predicted labels that were then compared with the actual labels.

H. Particle Swarm Optimization for K Optimization

The optimization of the value of k was carried out using the Particle Swarm Optimization algorithm through the GlobalBestPSO class in the PySwarms library. The PSO configuration presented in TABLE 3 covers 10 particles ($n_particles=10$) with one search dimension ($dimensions=1$) that represents the candidate value of k , a cognitive coefficient $c1=1.5$, a social coefficient $c2=1.5$, an inertia weight $w=0.7$, a search range (bounds) between 1 and 50, and 20 optimization iterations. The objective function that was minimized is the value of 1 minus the mean accuracy of 5-fold cross-validation on the training data for each candidate value of k , so the smaller the value of the objective function, the higher the mean cross-validation accuracy obtained. It should be emphasized that PSO in this study is used purely to optimize the parameter k in KNN, without involving feature selection or attribute weighting. The execution results show that the optimization process converged after 20 iterations with a best value of k ($best_k$) of 4 and a best cost of 0.188125, as summarized in TABLE 4. The value $k=4$ was then used to retrain the KNN model on all of the training data and to evaluate it on the same testing data as the baseline model with $k=5$, so that the comparison of the two models can be made objectively. Because the search range considered by

PSO covers only 50 integer candidate values of k (1 to 50), an exhaustive or grid search over this range could, in principle, also identify the optimal value directly at a comparable computational cost. PSO was adopted here mainly for methodological consistency with prior applications of PSO-based hyperparameter optimization in sentiment analysis research [5], [10], rather than because of a demonstrated advantage in speed or optimum quality over exhaustive search on this specific one-dimensional problem; a direct comparison between PSO and exhaustive search was not carried out in this study.

TABLE 3. Particle Swarm Optimization Parameter Configuration

Parameter	Value
Number of particles ($n_particles$)	10
Search dimensions ($dimensions$)	1 (value of k)
Cognitive coefficient ($c1$)	1.5
Social coefficient ($c2$)	1.5
Inertia weight (w)	0.7
Search range (bounds)	1 – 50
Number of iterations ($iters$)	20
Objective function	1 – mean 5-fold cross-validation accuracy

I. Model Evaluation

The performance evaluation of both models, namely the baseline KNN and KNN-PSO, was carried out using a classification report and a confusion matrix that produce accuracy, precision, recall, and F1-score values for each sentiment class. Accuracy measures the proportion of correct predictions against the total testing data, as formulated in (1). Precision measures the exactness of the model in predicting a class out of all data predicted as that class, as in (2). Recall measures the ability of the model to detect the data of a class out of all data that actually belongs to that class, as in (3). F1-score is the harmonic mean of precision and recall, as in (4), which is useful for assessing the balance of model performance across classes with unequal amounts of data.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

$$\text{Precision} = TP / (TP + FP) \quad (2)$$

$$\text{Recall} = TP / (TP + FN) \quad (3)$$

$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (4)$$

IV. RESULTS AND ANALYSIS

A. Data Collection Results

The scraping process using Google Play Scraper over the range 1 June 2026 to 1 September 2026 successfully collected 5,000 reviews in line with the target set in Section III-B. After the removal of duplicate data based on identical review content, 2,959 unique reviews remained, which means that around 40.8% of the raw data consisted of repeated reviews such as “bagus”, “mantap”, or “keren” that are commonly found in application reviews with high ratings. The amount of data at each processing stage is summarized in TABLE 2.

B. Text Preprocessing Results

The preprocessing stage on the 2,959 unique reviews produced gradual changes in the text representation, starting from the cleaning of irrelevant characters, case folding, the normalization of non-standard words using a reference dictionary, tokenization, and finally stopwords removal. After all of those stages, rows with empty (missing) values in any of the preprocessing columns were removed using dropna, leaving 2,858 labeled reviews ready for the modeling stage, as shown in Table 2. The difference of 101 records between the amount after duplicate removal (2,959) and the amount after preprocessing and labeling (2,858) shows that a small portion of reviews became empty after passing through cleaning and stopwords removal, for example reviews that contained only emoji, links, or words that all belonged to the stopwords list. The preprocessing stage does not only clean the text but also ensures that the data used in the classification process has a representation suitable for the model.

C. Sentiment Labeling Results

Based on the rating score, the sentiment data is divided into three classes with an imbalanced distribution, as presented in Fig. 3. Out of 2,858 labeled reviews, the Positive class dominates with 2,216 records (77.54%), followed by the Negative class with 522 records (18.26%), while the Neutral class only has 120 records (4.20%). This dominance of positively rated reviews is consistent with the findings of similar research on generative AI application reviews [14], [15], which also reported a tendency for users to give high ratings to artificial intelligence assistant applications. The proportion of the Neutral class of only around 4% of the whole data is an early indication that this class is potentially difficult to recognize by the classification model, because the amount of training data in that class is far lower than in the other two classes.

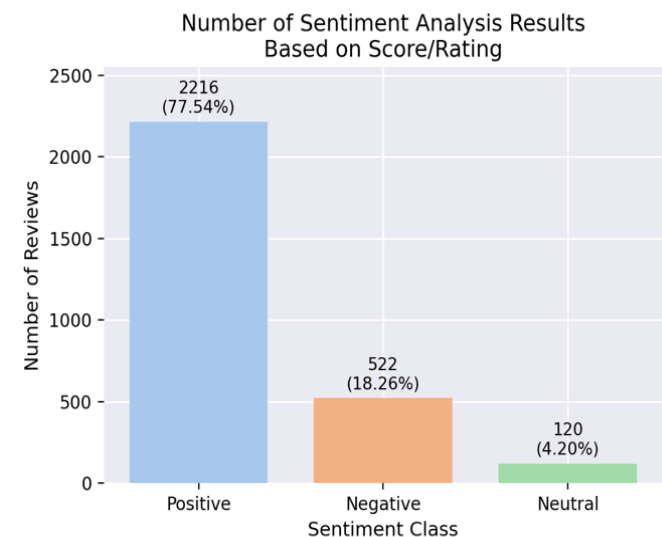


Fig. 3. Sentiment Class Distribution Based on Rating Scores

D. Feature Representation Results

The result of feature extraction using CountVectorizer on the 2,000 modeling records (after being split into 1,600 training and 400 testing samples as reflected in Fig. 2) produced 2,282 features, that is, the number of unique words forming the vocabulary of the whole training data. The relatively large number of features compared with the amount of training data (1,600 documents) shows that the bag-of-words representation in this study is sparse, because on average each review consists of only a few words after stopwords removal, while the feature space covers thousands of unique words from the whole corpus.

E. K-Nearest Neighbor Classification Results

The baseline KNN model with k=5 was trained using 1,600 vectorized training samples and tested on 400 testing samples. This model obtained an accuracy of 0.795 or 79.50%. The precision, recall, and F1-score values for each sentiment class are presented in Table 5, while the confusion matrix is shown in Fig. 4. The confusion matrix shows that out of 75 testing samples of the Negative class, the model only classified 26 samples correctly, while 47 samples were classified as Positive and 2 samples as Neutral. Out of 16 testing samples of the Neutral class, not a single one was classified correctly: 4 samples were predicted as Negative and 12 samples were predicted as Positive. In contrast, out of 309 testing samples of the Positive class, 292 samples were classified correctly, while the rest were spread into the Negative class (14 samples) and the Neutral class (3 samples). This pattern confirms that the baseline model is very reliable in detecting the Positive class (recall 0.945) but weak on the Negative class (recall 0.347) and completely fails on the Neutral class (recall 0.000).

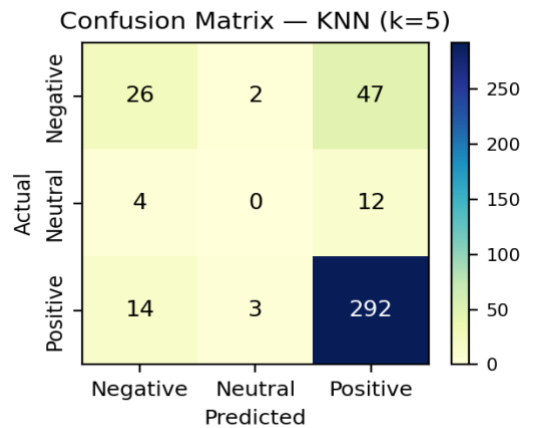


Fig. 4. Confusion Matrix of the KNN Model (K=5)

F. Particle Swarm Optimization Results

The PSO optimization process was run with 10 particles and 20 iterations as explained in Section III-H, searching for a value of k in the range 1 to 50 that minimizes the value of 1 minus the mean accuracy of 5-fold cross-validation on the training data. The optimization process converged at the 20th iteration with a best cost of 0.188125 and a best particle position of 3.974, which after rounding produces best_k = 4. This finding shows that, based on the cross-validation estimate on the training data, the

KNN configuration with $k=4$ is estimated to give the highest mean cross-validation accuracy ($1 - 0.188125 = 0.811875$ or around 81.19%) compared with the other candidate values of k evaluated by PSO within the search range, including $k=5$ which was used as the baseline. A summary of the optimization results is presented in TABLE 4.

TABLE 4. Parameter K Optimization Results by PSO

PSO Output	Value
Best k (selected $n_neighbors$)	4
Best particle position (before rounding)	3.974
Best cost ($1 - \text{mean CV accuracy}$)	0.188125
Estimated CV accuracy at best k	0.811875 (81.19%)

G. KNN-PSO Classification Results

The KNN-PSO model with $k=4$ was then retrained on the same 1,600 training samples and evaluated on the same 400 testing samples as the baseline model, producing an accuracy of 0.8075 or 80.75%. The confusion matrix of this model is shown in Fig. 5. Compared with the baseline confusion matrix, the number of correct Negative predictions rose from 26 to 29, while the misclassification of Negative into Positive decreased from 47 to 43. In the Positive class, the number of correct predictions rose from 292 to 294, with errors into the Negative class decreasing from 14 to 13, while errors into the Neutral class decreased from 3 to 2. The Neutral class again had no correctly classified sample in the KNN-PSO model: the error pattern is almost identical to the baseline, namely 4 samples predicted as Negative and 12 samples predicted as Positive out of a total of 16 testing samples of the Neutral class.

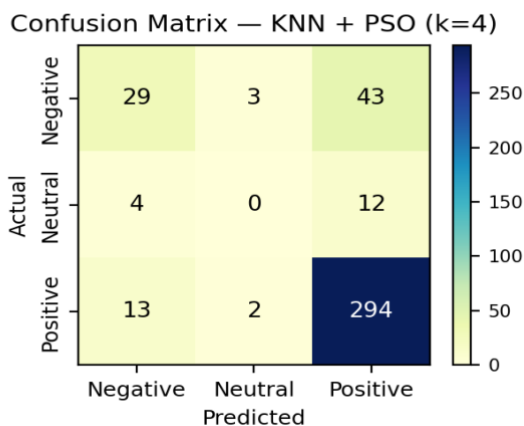


Fig. 5. Confusion Matrix of the KNN-PSO Model ($K=4$)

H. Comparison of KNN and KNN-PSO

A complete comparison between the baseline KNN model and the KNN-PSO model is summarized in TABLE 6 and visualized in Fig. 6 for accuracy and Fig. 7 for the F1-score of each class. Accuracy increased from 79.50% to 80.75%, or a rise of 1.25 percentage points. The macro F1-score, which gives equal weight to all three classes without taking the amount of data into account, increased from 0.441 to 0.458, while the weighted F1-score, which takes the data proportion of each class into account, increased from 0.765 to 0.780. At the class level,

the precision, recall, and F1-score of the Negative class increased quite clearly, from $F1=0.437$ to $F1=0.479$, while the Positive class increased slightly from $F1=0.885$ to $F1=0.894$. The Neutral class showed no change at all, with precision, recall, and F1-score remaining at 0.000 in both models, as presented in TABLE 5.

TABLE 5. Classification Report per Sentiment Class: KNN vs KNN-PSO

Class	Precision		Recall		F1-Score	
	KNN	KNN-PSO	KNN	KNN-PSO	KNN	KNN-PSO
Negative	0.591	0.630	0.347	0.387	0.437	0.479
Neutral	0.000	0.000	0.000	0.000	0.000	0.000
Positive	0.832	0.842	0.945	0.951	0.885	0.894

TABLE 6. Overall Evaluation Metrics Comparison of KNN and KNN-PSO

Metric	KNN ($K=5$)	KNN-PSO ($K=4$)
Accuracy	0.795 (79.50%)	0.808 (80.75%)
Macro F1-Score	0.441	0.458
Weighted F1-Score	0.765	0.780

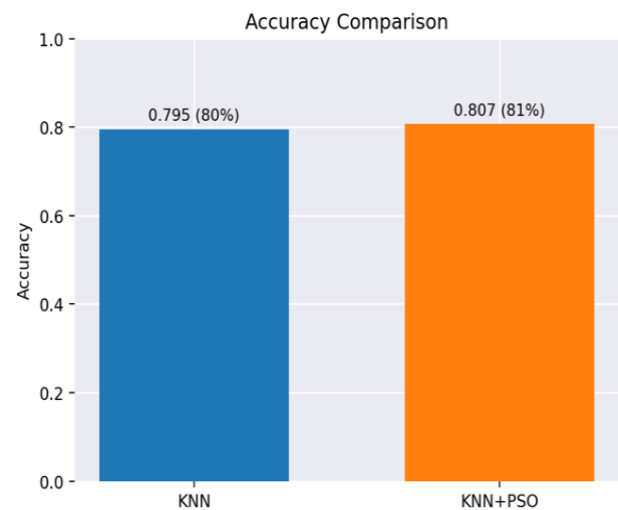


Fig. 6. Accuracy Comparison of the KNN and KNN-PSO Models

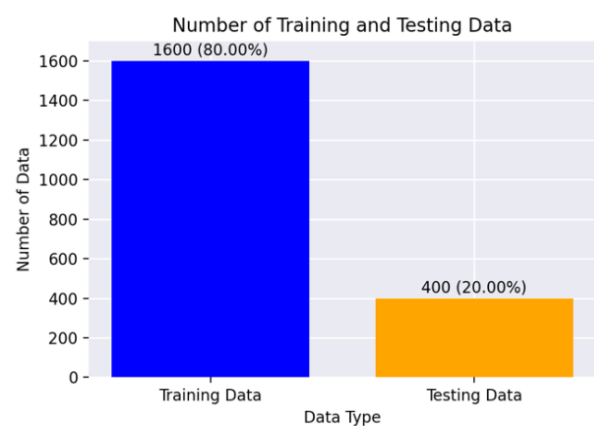


Fig. 7. F1-Score Comparison per Class: KNN vs KNN-PSO

I. Discussion

The change in accuracy of 1.25 percentage points is a relatively small improvement, and the term “significant” is not used in this study to describe that change because no statistical significance test was performed on the difference in accuracy or F1-score between the two models. What can be stated objectively is that PSO optimization produced a consistent improvement in accuracy, macro F1-score, and weighted F1-score, as well as in the precision, recall, and F1-score of the Negative and Positive classes, but that improvement has not resolved the problem of identifying the Neutral class. The failure of the model to recognize the Neutral class, both before and after optimization, is most likely related to the proportion of data in this class of only 4.20% of the whole labeled data as shown in Fig. 3, so the number of Neutral examples in the training data (estimated at around 67 out of 1,600 records based on the same proportion) is far smaller than the number of neighbors $k=4$ or $k=5$ used by KNN to decide the class of a test sample. In other words, for most test points that actually belong to the Neutral class, the nearest neighbors in the CountVectorizer feature space are most likely dominated by Positive or Negative data, so KNN systematically predicts one of those two classes.

The PSO objective function in this study purely maximizes the mean cross-validation accuracy, without considering metrics that are sensitive to class imbalance such as the macro F1-score. Because the Positive class dominates the data (77.54%), the value of k that maximizes overall accuracy tends to also be the value of k that is most favorable for the majority class, as can be seen from the increase in Positive recall from 0.945 to 0.951 after optimization. This pattern is consistent with the working mechanism of KNN itself: a smaller value of k ($k=4$ compared with $k=5$) makes the classification decision more influenced by the closest neighbors, which on data dominated by the Positive class most likely still come from the Positive class, so this optimization implicitly benefits the majority class and the class with a sufficiently distinct textual pattern (Negative), but gives no benefit to the Neutral class whose data is too small to form a coherent neighborhood.

The decision to remove the words “bug” and “error” at the stopword removal stage, as explained in Section III-C, is also relevant for explaining the performance of the Negative class, which, even though it improved, remains at a moderate level of precision and recall (precision 0.630 and recall 0.387 in the KNN-PSO model). The removal of those two terms potentially reduces the textual signal that explicitly marks a technical complaint, so the model has to rely on other words that are more ambiguous to recognize negative reviews, for example general adjectives that can also appear in a positive context.

Compared with research that applies PSO to optimize other classification algorithms rather than the parameter k of KNN [5], [10], the approach in this study has a narrower search space because only one dimension (the value of k) is optimized without touching the feature representation or attribute weights. This limitation of the search space is consistent with the size of the improvement obtained, namely only 1.25 percentage points in accuracy, far smaller than the potential improvement usually

reported in research that combines parameter optimization with feature selection or class balancing. On the other hand, an approach that purely optimizes k offers the advantage of a simple interpretation: every change in performance observed in Table 5 and Table 6 can be linked directly to the change of k from 5 to 4, without having to consider the combined effect of changes in the feature representation. Akbar [15] showed that adding SMOTE and Random Oversampling in the case of sentiment classification of three generative AI applications can help handle the dominance of positive sentiment; given that the Neutral class in this study was not detected at all either before or after PSO optimization, such a class balancing technique appears to have more potential to solve this problem than optimizing the parameter k alone.

The relevance of the number of features (2,282) to KNN performance also needs to be examined. The ratio of 2,282 features to 1,600 training documents produces a feature space larger than the number of samples, a condition that is generally related to the high-dimensionality problem in distance-based algorithms such as KNN, namely that the Euclidean distance between documents becomes less informative when the feature dimension is very high and most of the values are zero (sparse). The pattern of far better performance on the Positive class compared with the Negative and Neutral classes in this study is in line with the pattern commonly reported in KNN-based sentiment classification research on other application reviews [3], [7], although this study did not directly test whether performance would improve with a lower-dimensional feature representation.

Compared with research using non-KNN algorithms on generative AI application reviews, Rosandi et al. [14] reported that Support Vector Machine gave the highest sentiment classification performance on Gemini AI reviews compared with Naive Bayes and Random Forest. Given that KNN was not included in that comparison, this study fills that gap by placing KNN, both in its baseline form and as the result of PSO optimization, as an additional point of comparison in the domain of Google Gemini reviews. The accuracy of 79.50–80.75% obtained in this study is in a range comparable to several other KNN-based application review sentiment classification studies [3], [7], although a direct comparison is difficult to make because of differences in dataset, number of classes, and the training-testing data proportion in each study.

J. Research Limitations

The PSO configuration used, namely 10 particles and 20 iterations, is relatively small for a metaheuristic search. Although the optimization process in this study has shown signs of convergence (the best cost was stable approaching the 20th iteration), a larger configuration could explore the range $k=1-50$ more thoroughly, and it cannot be ruled out that another value of k in that range gives a comparable or better result than $k=4$. That condition needs to be taken into account as a methodological limitation in interpreting the optimization results obtained.

This study also does not apply any class-balancing technique, such as oversampling, undersampling, or SMOTE, to

the training data, and the KNN and PSO configurations were evaluated only on the naturally imbalanced distribution described in Section IV-C. This design choice keeps the analysis focused on isolating the effect of the PSO-optimized parameter k from the effect of any resampling strategy, but it also means that the reported results do not indicate whether a balancing technique could improve the classification of the Neutral class; that question is left open for future work rather than addressed empirically in this study. In addition, the 2,000 records used for modeling were taken as the first, and therefore most recent, rows of the 2,858 preprocessed reviews rather than through random sampling, as noted in Section III-F; because this selection follows the chronological order preserved from scraping, it may not fully represent the temporal or topical variation present in the complete preprocessed dataset.

Furthermore, the comparison between the baseline KNN and KNN-PSO models in this study is based on a single train-test split with a fixed random state=42. Repeated cross-validation or evaluation across multiple random splits, which would allow the stability of the 1.25-percentage-point improvement to be expressed as a mean and standard deviation, was not performed in this study, so the robustness of this improvement to different data partitions has not been established.

Overall, the results of this study show that PSO optimization of the KNN parameter k gives a real but limited improvement in the sentiment classification of Google Gemini reviews: accuracy, macro F1-score, and weighted F1-score all increased, and the performance of the Negative class improved more than that of the Positive class, which was already high in the baseline. However, this optimization does not touch the root of the problem in the Neutral class, which still has precision, recall, and F1-score of 0.000 in both models. This finding confirms that an increase in accuracy alone is not enough to judge the quality of a sentiment classification model in a case with three imbalanced classes, and that the macro F1-score, which is far lower (0.441–0.458) than the weighted F1-score (0.765–0.780) and the accuracy (79.50–80.75%), is an important indicator that needs to be reported clearly so that the performance of the model on the minority class can be evaluated without relying on the overall accuracy value.

V. CONCLUSION

This study applied the K-Nearest Neighbor algorithm to classify the sentiment of Indonesian-language Google Gemini user reviews into three classes, namely Negative, Neutral, and Positive, then optimized the value of the parameter k using Particle Swarm Optimization to evaluate its effect on classification performance. The scraping process produced 5,000 reviews over the range 1 June to 1 September 2026, which after duplicate removal, preprocessing, and labeling left 2,858 records, of which 2,000 were used as modeling data and split into 1,600 training and 400 testing samples. Feature representation using CountVectorizer produced 2,282 features. The baseline KNN model with $k=5$ obtained an accuracy of 79.50%, while PSO with 10 particles and 20 iterations selected $k=4$ based on a best cost of 0.188125, so the KNN-PSO model obtained an accuracy of 80.75%, or an increase of 1.25

percentage points compared with the baseline. This improvement is also reflected in the macro F1-score, which rose from 0.441 to 0.458, and the weighted F1-score, which rose from 0.765 to 0.780, contributed mainly by the improved performance of the Negative class. Even so, the Neutral class remained completely undetected by both models, with precision, recall, and F1-score of 0.000, so PSO optimization in this study does not solve all classification problems but only improves performance on classes that already have adequate data representation. The contribution of this study lies in applying PSO purely for the optimization of the parameter k without involving feature selection, so the effect of the optimization on KNN performance can be interpreted directly and transparently, including its limitation with regard to a sentiment class that has very little data. The limitations of this study include the use of a subset of 2,000 records out of the 2,858 records resulting from preprocessing, the imbalance in the amount of data between classes, and a PSO objective function that only improves overall accuracy without considering the balance between classes. Further research is suggested to consider a PSO objective function based on the macro F1-score, the application of class balancing techniques such as SMOTE or oversampling on the Neutral class as applied in other application sentiment classification research [15], [17], the use of all 2,858 preprocessed records for modeling, and an evaluation of the effect of keeping technical terms such as “bug” and “error” at the stopword removal stage on the classification accuracy of the Negative class.

VI. REFERENCES

- [1] Gemini Team, Google DeepMind, “Gemini: A Family of Highly Capable Multimodal Models,” Technical Report, Google DeepMind, 2023.
- [2] P. Perini and A. Susi, “Analysing App Reviews for Software Engineering: A Systematic Literature Review,” *Empirical Software Engineering*, vol. 27, no. 2, art. 43, 2022, doi: 10.1007/s10664-021-10065-7.
- [3] G. A. Ramadhan, M. Afdal, I. Permana, and M. Jazman, “Analisis Sentimen pada Ulasan Aplikasi Maxim di Google Play Store dengan K-Nearest Neighbor,” *JURIKOM (Jurnal Riset Komputer)*, vol. 10, no. 3, pp. 715–724, 2023, doi: 10.30865/jurikom.v10i3.6396.
- [4] M. R. Hanafi and R. K. R., “Analisis Sentimen pada Ulasan Aplikasi Sirekap di Google Play Menggunakan Algoritma Naive Bayes,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 4, pp. 1578–1586, 2024, doi: 10.57152/malcom.v4i4.1693.
- [5] V. Vamilina and R. Novita, “Analisis Sentimen E-Wallet Menggunakan Support Vector Machine Berbasis Particle Swarm Optimization,” *Building of Informatics, Technology and Science (BITS)*, vol. 5, no. 1, pp. 40–48, 2023, doi: 10.47065/bits.v5i1.3526.
- [6] D. N. Novianti, D. F. Shiddieq, F. F. Roji, and W. Susilawati, “Komparasi Algoritma Support Vector Machine dan Naive Bayes untuk Analisis Sentimen pada Metaverse,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 1, pp. 231–239, 2024, doi: 10.57152/malcom.v4i1.1061.
- [7] L. A. Susanto, “Komparasi Model Support Vector Machine dan K-Nearest Neighbor pada Analisis Sentimen Aplikasi Polri Super App,” *Jurnal Informatika dan Teknik Elektro Terapan (JITET)*, vol. 12, no. 2, pp. 1180–1190, 2024, doi: 10.23960/jitet.v12i2.4152.
- [8] S. A. R. Ansori and A. N. Fajar, “Studi Komparatif Algoritma KNN, SVM, dan Naive Bayes untuk Analisis Sentimen pada Review Google Play Aplikasi Traveloka,” *JITU: Journal of Informatics Technology and Communication*, vol. 9, no. 2, pp. 192–201, 2025, doi: 10.36596/jitu.v9i2.1861.

- [9] S. A. Putri and K. Kusnawi, "Komparasi Algoritma Naive Bayes, KNN, dan SVM pada Analisis Sentimen Ulasan Aplikasi Media Sosial di Google Play Store," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 7, no. 3, pp. 602–610, 2023, doi: 10.29207/resti.v7i3.4883.
- [10] W. Nugraha and M. Hardjianto, "Optimasi Support Vector Machines (SVM) Menggunakan Particle Swarm Optimization (PSO) pada Analisis Sentimen Ulasan Pengguna Layanan Bukalapak di Twitter," *Decode: Jurnal Pendidikan Teknologi Informasi*, vol. 4, no. 3, pp. 1082–1100, 2024, doi: 10.51454/decode.v4i3.686.
- [11] F. Nufairi, N. Pratiwi, and F. Herlando, "Analisis Sentimen pada Ulasan Aplikasi Threads di Google Play Store Menggunakan Algoritma Support Vector Machine," *JIPi (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 9, no. 1, pp. 339–348, 2024, doi: 10.29100/jipi.v9i1.4929.
- [12] N. D. Kurniawan, P. R. Ferdian, and N. Hidayati, "Analisis Sentimen Algoritma Naive Bayes, Support Vector Machine, dan Random Forest pada Ulasan Aplikasi Ajaib," *Jurnal Nasional Teknologi dan Sistem Informasi (TEKNOSI)*, vol. 11, no. 1, pp. 87–97, 2025, doi: 10.25077/TEKNOSI.v11i1.2025.87-97.
- [13] I. Rahmawati, T. R. Fitriani, and A. Y. P. Yusuf, "Analisis Sentimen Menggunakan Algoritma Naive Bayes pada Aplikasi m-BCA Berdasarkan Ulasan Pengguna di Google Play Store," *Jurnal Riset Informatika dan Teknologi Informasi*, vol. 1, no. 2, pp. 38–42, 2024, doi: 10.58776/jriti.v1i2.116.
- [14] R. Rosandi, A. I. Febrianto, A. A. Gibran, W. Bismi, I. Kurniawati, and R. Fahlapi, "Analisis Sentimen Masyarakat terhadap Penggunaan Gemini AI dengan Metode Machine Learning," *Djtechno: Jurnal Teknologi Informasi*, vol. 6, no. 3, pp. 1105–1117, 2025, doi: 10.46576/djtechno.v6i3.7962.
- [15] M. F. Akbar, "Analisis Sentimen Pengguna terhadap Aplikasi AI di Google Playstore Menggunakan Algoritma Support Vector Machine dan Naive Bayes (Studi Kasus: ChatGPT, Google Gemini, dan Microsoft Copilot)," *Skripsi S1, Universitas Mercu Buana, Jakarta*, 2025.
- [16] M. I. Raif, N. N. Hidayati, and T. Matulatan, "Otomatisasi Pendeteksi Kata Baku dan Tidak Baku pada Data Twitter Berbasis KBBI Menggunakan Metode Lexicon-Based," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK)*, vol. 11, no. 2, pp. 337–348, 2024, doi: 10.25126/jtiik.20241127404.
- [17] D. F. Nawulansih, N. C. Santi, and I. A. Sa'ida, "Analisis Sentimen Ulasan Aplikasi DANA di Google Play Store: Penerapan Support Vector Machine dan Synthetic Minority Over-sampling Technique," *Jurnal Pendidikan dan Teknologi Indonesia*, vol. 5, no. 9, pp. 2660–2671, 2025, doi: 10.52436/1.jpti.1053.

This page is intentionally left blank