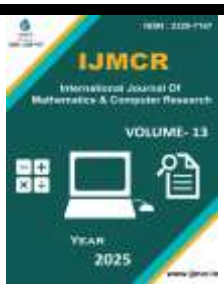
International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 13 Issue 05 May 2025, Page no. – 5260-5266

Index Copernicus ICV: 57.55, Impact Factor: 8.615

DOI: 10.47191/ijmcr/v13i5.18



Sentiment Classification of Livin by Mandiri App Reviews with Support Vector Machine Based on Modified Particle Swarm Optimization

Salsa Febriliana Sandita[1], Iut Tri Utami[2], Masithoh Yessi Rochayani[3]

[1,2,3]Department of Statistics, Faculty of Science and Mathematics, Diponegoro University

ARTICLE INFO

Published Online:
29 May 2025

Corresponding Author:
Iut Tri Utami

ABSTRACT

This research aims to evaluate the quality of the Livin by Mandiri mobile banking application based on user reviews from the Google Play Store. A Support Vector Machine (SVM) classifier is applied to distinguish between positive and negative sentiments. Modified Particle Swarm Optimization (MPSO) is used for parameter tuning because SVM performance is sensitive to parameter selection. The baseline SVM attains 90.30% accuracy, 90.64% precision, 89.86% recall, and 90.25% F1-score with a linear kernel with C = 1 and a 90:10 training-testing split. Accuracy is increased to 91.33% by the SVM-MPSO model, with precision of 90.87%, recall of 91.71%, and F1-Score of 91.29%. The Streamlit framework for interactive sentiment classification is used to deploy the model.

KEYWORDS: Livin by Mandiri; Sentiment Analysis; Support Vector Machine; Modified Particle Swarm Optimization; Streamlit

I. INTRODUCTION

The rise in internet connectivity and global digitalization has profoundly altered public behavior, particularly within the financial industry. In Indonesia, internet penetration reached 79.5% in 2024 [1], contributing to a 9.1% year on year increase in digital banking transactions [2]. These developments have spurred innovation across financial services, mostly notably the growing adoption of mobile banking as a convenient and efficient platform for routine banking activities.

Mobile banking platforms, exemplified by Livin by Mandiri, play a pivotal role in this digital transformation. Beyond basic functionalities such as fund transfers and bill payments, the application integrates advanced features most notably investment management tools that address customers expanding needs amid economic uncertainties, including inflation. While such innovations attract new users, they have also generated mixed feedback, with some customers expressing concerns regarding performance and service quality. Consequently, a systematic evaluation of service quality is essential for assessing user satisfaction and identifying areas for improvement.

To this end, the present research employs sentiment analysis based on text mining. Support Vector Machine (SVM) is widely adopted for sentiment classification due to its capability to handle high-dimensional data [3]. Nevertheless, SVM requires extensive parameter tuning to achieve optimal performance. Particle Swarm Optimization (PSO) has been utilised to automate this process [4], yet it converage prematurely and become trapped in local optima. Therefore, this research adopts a Modified Particle Swarm Optimization (MPSO) scheme that incorporates an inertia weight mechanism to balance exploration and exploitation during parameter search, thereby aiming to improve classification accuracy.

This research analyzes the performance of the SVM algorithm optimized with Modified Particle Swarm Optimization (MPSO) in classifying user sentiment from reviews of the Livin by Mandiri application on the Google Play Store. The implementation is carried out in Python and integrated with Streamlit to develop a web based application, enabling a more user-friendly and interactive interface. The results of this research are expected to improve classification accuracy and provide valuable insights for developers in enhancing the service quality of the Livin by Mandiri application.

II. METHODOLOGY

The data used in this study consist of primary qualitative data in the form of user reviews of the *Livin' by Mandiri* application, collected from the Google Play Store at the following URL: https://play.google.com/store/apps/details?id=id.bmri.livin&hl=id. Data collection was conducted using a web scraping technique implemented in the Python

programming language via the Google Colaboratory environment. A total of 4981 user reviews were collected during the period from October 9 to November 9, 2024.

This research involves both independent and dependent variables. The independent variable is the textual content of user reviews, while the dependent variable is user sentiment, which is categorized into two classes: positive and negative.

Several software tools were employed throughout the study. Python was utilized for data processing and sentiment analysis; Microsoft Excel was used for preliminary data exploration and structuring; and the Streamlit framework was adopted to present the analysis results in the form of an interactive web-based application.

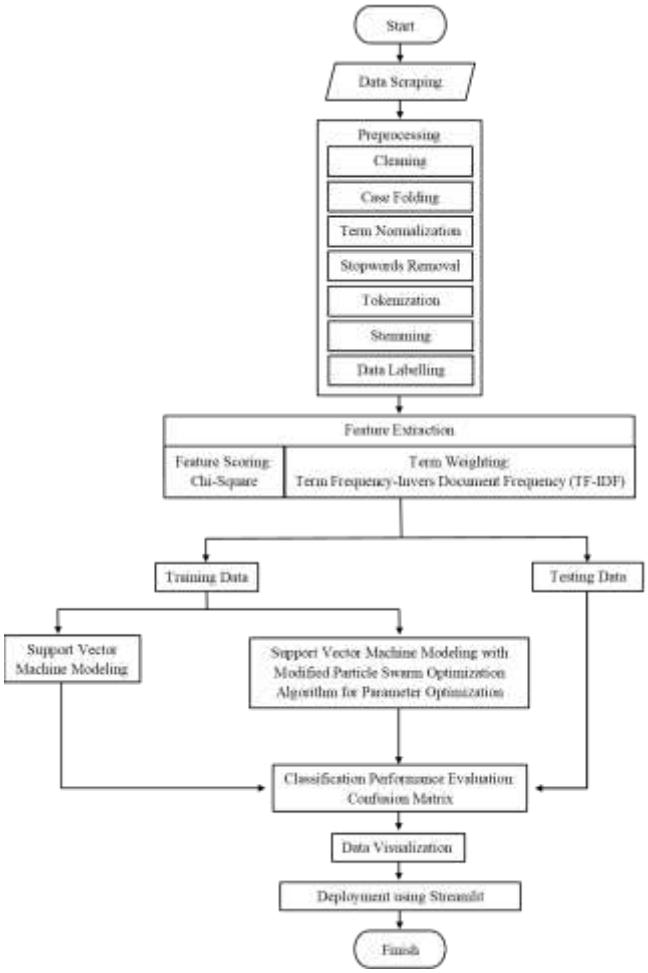


Figure 1. Analyze Data

A. Preprocessing

Data preprocessing was conducted to transform unstructured user review data into a more structured and analyzable format. The process began with data cleaning, which involved the removal of irrelevant elements such as duplicate entries, numerical characters, excess whitespace, punctuation marks, and emojis to improve data quality [5]. This was followed by case folding, where all text was converted to lowercase to ensure consistency [6].

Subsequently, word normalization was performed by converting non-standard words, abbreviations, and foreign terms into standard Indonesian words based on the official dictionary. The next steps included stopword removal to eliminate contextually insignificant words, tokenization to split the text into individual tokens, stemming to reduce words to their root forms, and finally, data labeling to categorize the sentiment of each review [7].

B. Feature Extraction

Chi-Square feature scoring is then applied to feature extraction in order to select the most significant features [7]. The Chi-Square test determines whether a word's occurrence is independent of its sentiment category in cases involving binary classification, such as sentiment analysis (positive vs. negative). Since both contribute equally to the overall association strength, the Chi-Square score for a word in the positive class will be equal to the score in the negative class when the dataset is balanced between the positive and negative classes. These Chi-Square scores are then assessed using a p-value. the word has a statistically significant association with the sentiment class and can be regarded as an important feature if the corresponding p-value is below the threshold. The following Chi-Square value was obtained from equation (1):

$$\chi^2(t,c) = \frac{N(A_cD_c - C_cB_c)^2}{(A_c+C_c)(B_c+D_c)(A_c+B_c)(C_c+D_c)} \quad (1)$$

The selected features are then weighted using Term Frequency-Inverse Document Frequency (TF-IDF) to determine the importance of each word in the document [7]. According [8], TF-IDF weighting is formulated as in Equation (2):

$$W_{j,i} = \frac{n_{j,i}}{\sum_{j=1}^{k} n_{j,i}} \times log_2 \frac{D}{d_j} \quad (2)$$

C. Mining Process

Data that has been preprocessed and feature extraction is then divided by a ratio of 90:10 ration and classification analyze. Sentiment classification can be done with the Support Vector Machine (SVM) method. SVM is a supervised learning method that focuses on finding the optimal hyperplane to separate two classes of data [9]. In the case of binary classification, the separation of positive (+1) and negative (-1) classes is determined by Equation (3):

$$(\boldsymbol{w^T x_i}) + b = 0 \quad (3)$$

Optimization in SVM aims to maximize the margin between classes by solving the quadratic problem in Equation (4):

$$min \frac{1}{2}\|\boldsymbol{w}\|^2, \text{ with } \|\boldsymbol{w}\|^2 = \boldsymbol{w^T w} \quad (4)$$

with the constraint $y_i(\boldsymbol{w^T x_i} + b) \geq 1 \;\; ; i = 1,2,\dots,n$

The quadratic programming method optimization in Equation (4) is more easily solved by the Lagrange Multiplier computational technique. The optimal solution is obtained by forming a Lagrange function as formulated in Equation (5):

$$L_p(w,b,\alpha) = \frac{1}{2}\boldsymbol{w^T w} - \Sigma_{i=1}^{n}\alpha_i[y_i(\boldsymbol{w^T x_i} + b) - 1] \quad (5)$$

The value α_i is the Lagrange Multiplier coefficient associated with x_i and is zero or positive ($\alpha_i \geq 0$), but for other points the value of $\alpha_i = 0$ [10]. The coefficient α_i is used to

determine w and the optimal value is obtained by calculating the first derivative of w and b.

The first derivative of w is formulated as in Equation (6):

$$\frac{\partial L_p}{\partial \boldsymbol{w}} = \mathbf{0} \rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i} = \mathbf{0} \quad (6)$$

The first derivative of b is formulated as in Equation (7):

$$\frac{\partial L_p}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \quad (7)$$

The first derivative transformation of the Primal Lagrangian function generates a relationship between the weight vector w, the variable α_i, and the training data $\mathbf{x_i}$ because the value of α_i is unknown, so the values of w and b cannot be directly determined. The direct solution is difficult to solve, especially if the number of features used is very large. In overcoming this, the Lagrange Multiplier method is used to convert into a dual problem form as in Equation (8):

$$L_D(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^T \boldsymbol{x}_j) \quad (8)$$

with constraints $\Sigma_{i=1}^{n} \alpha_i y_i = 0 \;\; ; \alpha_i \geq 0$

The α_i values are used to determine the weights w contained in Equation (5). Each training data has an α_i value, with $\alpha_i > 0$ value indicating a support vector, while $\alpha_i = 0$ indicates that the data is not located on the hyperplane. The resulting decision function is only influenced by the support vector. The classification function (hyperplane) is obtained using the formula in Equation (9):

$$(z) = sign\left(\sum_{i=1}^{s} \alpha_i y_i (\boldsymbol{x}_i^T \boldsymbol{x}_j) + b\right) \quad (9)$$

Approach in SVM is not only used to separate data for optimization, but also allows for more efficient calculations without the need to perform an explicit transformation space, making it more effective for handling high-dimensional data [11]. The kernel approach works by mapping the data x_i using function $\phi(\mathbf{x})$, so that the dot product between two vectors in the feature space can be formulated as Equation (10):

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \quad (10)$$

In SVM, the determination of the support vector depends only on the kernel function without the need to know the form of the mapping function ϕ [12]. The hyperplane separation function generated after SVM training is expressed as in Equation (11):

$$f(\phi(x)) = sign\left(\sum_{i=1}^{s} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + b\right) \quad (11)$$

some kernel functions that are often used for classification in sentiment analysis can be seen in Table 1.

Table 1. Type of Kernel

Kernel Type	*Formula*
Linear	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
Polynomial	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + h)^d$
Radial Basis Function (RBF)	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$

The SVM classification stage begins with preprocessing text data which is then converted into numerical representations using Chi-Square and TF-IDF. After that, the training and testing data are used to build the SVM classification model by determining the optimal hyperplane that separates the positive and negative classes. The separation is done by maximizing the margin between the data points of both classes using an appropriate kernel function. Next, the SVM model is evaluated based on accuracy, precision, recall, and F1-Score metrics, which are calculated using a confusion matrix based on the number of correct and incorrect predictions.

Classification performance can be improved by parameter optimization using Modified Particle Swarm Optimization (MPSO), which aims to find the optimal C value so that the model can have more generalization to new data [13]. The SVM-MPSO classification process begins with parameter initialization with the C value randomly selected within a certain range. After that, each value of C is tested on the SVM model and performance is measured based on model accuracy. In each iteration, MPSO updates the particle velocity and position using the formulas contained in Equation (12) and Equation (13):

$$v_i^{t+i} = \rho v_i^t + c_1 r_1 (pBest_i^t - x_i^t) + c_2 r_2 (gBest_g^t - x_i^t) \quad (12)$$

$$x_i^{t+i} = x_i^t + v_i^t \quad (13)$$

Where pBest is the best local position, and gBest is the best global position. The balance between exploration and exploitation is ensured through the inertia weights in the MPSO updated using Equation (14):

$$\rho_t = \rho_{max} - \frac{\rho_{max} - \rho_{min}}{t_{max}} \times t \quad (14)$$

After the iteration is completed, the optimal C value is obtained from gBest and applied in the SVM model. Optimization allows the model to produce higher accuracy in sentiment classification compared to SVM without optimization. Evaluation of model performance is done using the confusion matrix formulated as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (15)$$

$$Precision = \frac{\left(\frac{TP}{TP+FP}\right)+\left(\frac{TN}{TN+FN}\right)}{2} \quad (16)$$

$$Recall = \frac{\left(\frac{TP}{TP+FN}\right)+\left(\frac{TN}{TN+FP}\right)}{2} \quad (17)$$

$$F1-Score = 2\left(\frac{Precision \times Recall}{Precision+Recall}\right) \quad (18)$$

The results of this sentiment classification are then visualized using bar charts to see the frequency of reviews that are often discussed in positive and negative sentiments [14]. Once visualized, the sentiment classification is implemented in a Streamlit based application. Streamlit is a Python based framework for building interactive visual interfaces quickly and efficiently, making it easy for users in various circles to perform real-time sentiment analysis with an intuitive display [15].

III. RESULT AND DISCUSSION

Data was obtained through scraping techniques of scraping and successfully collected as much as 4981 review data. After

preprocessing, as much as 1064 of the data was eliminated, leaving 3917 of the review data ready for further analysis. Of these, the labeling process results in 2071 negative reviews and 1848 positive reviews. The yield of preprocessing data is shown in Table 2.

Table 2. Data Preprocessing

No.	*Before Preprocessing*	*After Preprocessing*
1	Banyak manfaat Mudah bertransaksi dn cepat..	banyak manfaat mudah transaksi cepat
2	Tidak bisa menghapus yang Riwayat Transfer. Ã°Å¸â€˜Å½Ã° Å¸â€˜Å½Ã°Å¸â€˜Å½	tidak dapat hapus riwayat transfer
3	Aflikasi suka ngeleg g bisa di buka pas lagi butuh2 nya	aplikasi suka lag tidak dapat buka saat butuh

Following preprocessing, feature extraction which includes feature scoring and word weighting is applied to the data. Equation (1) is used to determine the $\chi 2(T)$ value for each word term in order to perform feature scoring using Chi-Square. The top 500 words with the highest Chi-Square value and a value higher than the p-value are chosen to be used in classification model construction. Table 2 displays the outcomes of the Chi-Square computation example.

Table 3. Chi-Square Process

Features (T)	χ^2(T)	*p-value*
tidak	713.88409	3.0499
update	388.49317	1.5929
verifikasi	254.46393	9.1773
…	…	…
center	3.030104	0.0817
sisa	3.018961	0.0823

The words "tidak," "update," and "verifikasi" have the highest Chi-Square values, according to Table 3. The words "center" and "sisa" have the lowest Chi-Square value, indicating that they are less relevant in determining the classification category, whereas these words are the most relevant in determining the classification category. Table 2 displays the words that have been given Chi-Square values. Here is an example of a manual Chi-Square calculation where values are assigned to each word:

$$\chi^2(t,c) = \frac{N(A_cD_c - C_cB_c)^2}{(A_c+C_c)(B_c+D_c)(A_c+B_c)(C_c+D_c)}$$

$$\chi^2\ (tidak, negative) = \frac{3917((310\times860)-1211\times1536))^2}{(310+1211)(1536+860)(310+1536)(1211+860)} = 713.884$$

$$\chi^2\ (tidak, negative) = \frac{3917((310\times860)-1211\times1536))^2}{(310+1211)(1536+860)(310+1536)(1211+860)} = 713.884$$

$$\chi^2\ (tidak, positive) = \chi^2\ (tidak, negative)\ = 713.884$$

The Term Frequency-Inverse Document Frequency (TF-IDF) method, which is implemented using the scikit-learn library in the Python programming language, is used in this study to carry out the word weighting process. By giving words weights according to their significance in each review, this method turns text data into numerical values. The dataset used in this process includes 4280 affixed words that were taken from user reviews. Of these, 500 words were chosen as key features for further analysis, and the TF-IDF method was used to weight these features. Below is a manual calculation example to show how TF-IDF operates.

$$W_{jelas,doc3} = \frac{number\ of\ feature\ "jelas"\ in\ doc\ 3}{all\ features\ "jelas"\ in\ doc\ 3} \times log_2 \frac{total\ number\ of\ doc}{number\ of\ doc\ contains\ features\ "jelas"} W_{jelas,doc3} = \frac{1}{4} \times log_2 \frac{3917}{122} = 1.2512$$

Classifying sentiment from reviews in the Livin' by Mandiri app using the Support Vector Machine (SVM) method. The team collected a total of 3917 reviews that had been labeled with sentiment in both the training and testing data. The testing data was used to validate its performance, while the training data was used to build the classification model. The proportion of data sharing was determined through trial and error, where several ratios were compared to determine the best accuracy, as shown in Table 5. In the end, the ratio used was 90% training data and 10% testing data. The classification model used SVM with a linear kernel, and the value of the regulation parameter C was determined using grid search, which included values of 0.01. 0.1. 1. 10. and 100. The model results are shown in Table 4.

Table 4. SVM Parameter

Parameter (T)	*Accuracy*
0.01	0.7500
0.1	0.8673
1	0.9031
10	0.8980
100	0.8827

The test results in Table 5 show that the most optimal C value is 1. because it produces the highest accuracy value of 90.31%. The SVM model is formed using the training data, by determining the support vector based on the dot product results that have been transformed to a higher dimensional space using a kernel function. In the training stage, the dot product process is performed by applying a linear kernel function to each pair of training data. The kernel function helps represent data through comparisons between pairs of data, rather than individually. An example of the results of calculating the linear kernel function on the top reviews in the training data is as follows.

Linear kernel function for review 1 and review 1:

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) = \boldsymbol{x}_1^T \boldsymbol{x}_1$$

$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) = (0.273 \times 0.273) + (0.283 \times 0.283) + \cdots + (0.408 \times 0.408) = 0.5133$

Kernel values are produced by applying the same calculation to all training data. Quadratic Programming (QP) is then used to determine the most optimal α_i and b values using the training data that the kernel has mapped. Due to the size of the data used, Google Collaboratory's Python software is used to calculate the values of α_i and b. The SVM hyperplane function with a linear kernel has an optimal bias value of b = 0.96053823, and the study's 1206 support vector data yields the following hyperplane equation for the SVM model with a linear kernel:

$f(\phi(x)) = sign\left(\sum_{i=1}^{s} \alpha_i y_i \kappa\left(\boldsymbol{x}_{training}, \boldsymbol{x}_{testing}\right) + 0.96053823\right)$

Using $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$ and S is the amount of data that becomes the support vector, $\boldsymbol{x}_i$ is the support vector and $\boldsymbol{x}_j$ is the testing data that needs to be predicted. Following the creation of the model using training data, the dot product between training and testing data is computed in a linear kernel function to perform the prediction process on testing data. For the 3513th training data review and the 393rd testing data, the following is an example of how to compute the linear kernel function between training and testing data

$\kappa(\boldsymbol{x}_{3513}, \boldsymbol{x}_{393}) = \boldsymbol{x}_{3513}^T \boldsymbol{x}_{393}$

$\kappa(\boldsymbol{x}_{3513}, \boldsymbol{x}_{393}) = (0 \times 0.463) + (0.285 \times 0.602) + \cdots + (0 \times 0.755) = 0.1717$

The computation continues until the kernel values for the 1206th training data and the 393rd testing data are determined. The data is assigned the class label +1 (positive) if $f(\phi(x))$ is positive, and -1 (negative) if it is negative. The following is the prediction for the 393rd testing data:

$f\left(\phi(x)\right) = sign\left(\sum_{i=1}^{s} \alpha_i y_i \kappa\left(x_{training}, x_{testing}\right) + 0.96053823\right)$

$f\left(\phi(x)\right) = sign\left(\sum_{i=1}^{s} \alpha_i y_i \kappa\left(x_{training}, x_{393}\right) + 0.96053823\right)$

$f\left(\phi(x)\right) = sign([(-0.2087 \times 0.45) + (-1.0 \times 0.33) + \cdots + (-0.9806 \times 0.69)] + 0.96053823)$

$f\left(\phi(x)\right) = sign(-0.823) = -1$

The 393rd testing data is categorized as class -1 (negative) based on the computation results. The optimal C parameter value, according to SVM classification performance with a linear kernel, is 1. The confusion matrix table is used to evaluate SVM classification performance with a linear kernel in the manner described below:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{151+203}{151+12+203+26} = 0.9031 = 90.31\%$$

$$Precision = \frac{\left(\frac{TP}{TP+FP}\right)+\left(\frac{TN}{TN+FN}\right)}{2} = \frac{\left(\frac{151}{151+12}\right)+\left(\frac{203}{203+26}\right)}{2} = \frac{0.926+0.887}{2} = 0.9064 = 90.64\%$$

$$Recall = \frac{\left(\frac{TP}{TP+FN}\right)+\left(\frac{TN}{TN+FP}\right)}{2} = \frac{\left(\frac{151}{151+26}\right)+\left(\frac{203}{203+12}\right)}{2} = \frac{0.853+0.944}{2} = 0.8986 = 89.86\%$$

$$F1-Score = 2\left(\frac{Precision \times Recall}{Precision+Recall}\right) = 2\left(\frac{0.9064 \times 0.8986}{0.9064+0.8986}\right) = 0.9025 = 90.25\%$$

Improving performance, parameter optimization is performed using the MPSO method. This technique finds the best parameters more efficiently than grid search. This optimization uses particles that represent a combination of parameters, with evaluation based on the fitness value or the best accuracy produced. In this stage, a set of initial parameters generated in the range of 0.01 to 100 is used as the initial data for the optimization process. MPSO acts as an optimization algorithm in the metaheuristic group. This algorithm is used to find more optimal SVM parameters to minimize errors in modeling. The stages of MPSO application are as follows:

1. MPSO Parameter Initialization
   The first stage in classification using the Modified Particle Swarm Optimization (MPSO) based Support Vector Machine (SVM) algorithm is parameter initialization. MPSO parameters are initialized randomly, including the number of particles, number of iterations, learning factors (c_1 and c_2), and inertia weights (ρ_{min} and ρ_{max}). The combination of these parameters is determined using Grid Search to obtain the optimal initial value. The MPSO parameters used can be seen in Table 5.

Table 5. MPSO Parameter

MPSO Parameter	*MPSO Parameter Values*
Number of Particle (N)	50
Number of Iteration	100
ρ_{max}	0.8
ρ_{min}	0.1
c_1 and c_2	1
r_1	1
r_2	0.5

2. Initialize position and velocity particle
   The initial position of the particle is randomly initialized within a predetermined range, while the initial velocity value of the particle = 0 because the particle has not moved
3. Determine the fitness value of the particle using the SVM algorithm
   The fitness value of the initial position of the particle is used to determine the best local position (pBest) and the best global position (gBest). In this context, fitness refers to the accuracy of the classification results obtained from the SVM algorithm with a linear kernel. Each C value produces a different level of accuracy.
4. Determining the best local position (pBest)

The pBest value for each particle C is determined based on the accuracy or fitness value generated by the SVM algorithm. This process aims to find the best local position. The pBest stage begins by comparing the pBest fitness value with the particle fitness value at the current iteration. If the pBest fitness value is greater than the particle fitness value at the current iteration, then the pBest position value will remain fixed. In the 1st iteration, the fitness value is equal to the pBest value.

5. Finding the best global position (gBest)
   The process of finding the best global position is done by determining the highest value of pBest. The highest pBest value is 0.8827, so the gBest in the 1st iteration is 0.8827.
6. Update the particle speed in the next iteration and do it repeatedly until it reaches the maximum iteration.

The results of parameter tuning with MPSO are shown in Table 6.

Table 6. Parameter of SVM-MPSO

Particle	*Cost (C)*	*Fitness (Accuracy)*
1	1.4470	0.9107
2	1.6559	0.9133
…	…	…
23	1.5289	0.9133
…	…	…
100	1.5752	0.9107

The highest fitness value is obtained at the 23rd particle with a parameter value of C = 1.528947. Based on the optimal C parameter value, the SVM model becomes more adaptive, so this method can avoid overfitting and improve overall classification accuracy. The SVM-MPSO model produces 1101 data as a support vector with a bias value of b = 1.0004635. The hyperplane equation obtained is as follows:

$$f(\phi(x)) = sign\left(\sum_{i=1}^{s} \alpha_i y_i \kappa\left(x_{training}, x_{393}\right) + 1.00046355\right)$$

All training data and the 393rd testing data are calculated until the kernel value is obtained. The data is assigned a class label of +1 if $f(\phi(x))$ is positive, and a class label of -1 if it is negative. The following is the prediction for the testing data from the 393rd:

$$f(\phi(x)) = sign\left(\sum_{i=1}^{s} \alpha_i y_i \kappa\left(x_{training}, x_{393}\right) + 1.00046355\right)$$

$$f(\phi(x)) = sign([(0.787885 \times 0.2280) + (1.694909 \times 0.2280) + \cdots + (-0.506958 \times 0.1927)] + 1.00046355)$$

$$f(\phi(x)) = sign(1.944) = +1$$

The performance evaluation of the sentiment classification model using the MPSO based SVM method shows the following results:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{153+205}{153+24+205+10} = 0.9133 = 91.33\%$$

$$Precision = \frac{\left(\frac{TP}{TP+FP}\right)+\left(\frac{TN}{TN+FN}\right)}{2} = \frac{\left(\frac{153}{153+24}\right)+\left(\frac{205}{205+10}\right)}{2} = \frac{0.864+0.953}{2} = 0.9087 = 90.87\%$$

$$Recall = \frac{\left(\frac{TP}{TP+FN}\right)+\left(\frac{TN}{TN+FP}\right)}{2} = \frac{\left(\frac{153}{153+10}\right)+\left(\frac{205}{205+24}\right)}{2} = \frac{0.939+0.895}{2} = 0.9171 = 91.71\%$$

$$F1-Score = 2\left(\frac{Precision \times Recall}{Precision+Recall}\right) = 2\left(\frac{0.9148 \times 0.9133}{0.9148+0.9133}\right) = 0.9129 = 91.29\%$$

The results show that the application of the MPSO algorithm can improve the accuracy of the classification model compared to the classification mode using only SVM, making it more effective in sentiment classification. The results of the SVM and SVM-MPSO model accuracy comparison are shown in Table 7.

Table 7. Result of SVM and SVM-MPSO

Model	*SVM*	*SVM-MPSO*
Accuracy	90.31%	91.33%
Precision	90.64%	90.87%
Recall	89.86%	91.71%
F1-Score	90.25%	91.29%

The SVM-MPSO model is then implemented in a Streamlit based application to facilitate interactive sentiment analysis. Streamlit is an open source Python framework that enables programmers to create interactive web applications from Python scripts without the need for extensive web technology expertise. After the sentiment classification model was successfully constructed and assessed, it was put into practice using this framework [15]. Whether a user review is positive or negative can be predicted by the developed application. The implementation of sentiment analysis using Streamlit is shown in Figure 2.

Figure 2. Streamlit Figure

The main view of the application is designed to be easy to use with a simple and intuitive interface. On the main page, users can find an input field to enter the review text they want to analyze. After the text is entered, the user only needs to press the "Analyze Sentiment" button to get the sentiment classification results. The application displays the prediction results with green text if the reviews entered are classified as

positive, while reviews classified as negative sentiment will be displayed with red text.

IV. CONCLUSIONS

Based on the discussion above and the result obtained, the following conclusions can be drawn. Sentiment classification using an SVM with a linear kernel and parameter C = 1 produced 90.31% accuracy, 90.46% precision, 89.86% recall, and 90.25% F1-Score. The MPSO optimization method enhanced performance, yielding an accuracy of 91.33%, precision of 90.78%, recall of 91.71%, and an F1-Score of 91.29% with an optimized parameter values of C = 1.528947. Technical problems like transaction failures, application instability, and login difficulties were the most common negative review topics. Positive reviews, on the other hand, emphasized the product's innovative features, appealing design, and ease of use. A Streamlit application was used to display the classification results, with red text denoting negative reviews and green text denoting positive reviews.

REFERENCES

1. Indonesian Internet Service Providers Association (APJII), "Indonesia's internet users reach 221 million people," APJII, [Online]. Available: https://apjii.or.id/berita/d/apjiijumlahpengguna-internet-indonesia-tembus-221-juta-orang. [Accessed: Oct. 5, 2024].
2. Financial Services Authority of Indonesia (OJK), "Indonesian banking statistics," [Online]. Available: https://www.ojk.go.id/id/kanal/perbankan/data-dan-statistik/statistik-perbankan-indonesia/default.aspx. [Accessed: Oct. 7, 2024].
3. C. Z. V. Junus, Tarno, and P. Kartikasari, "Classification using Support Vector Machine and Random Forest for early detection of diabetes mellitus risk," *J. Gaussian*, vol. 11. no. 3, pp. 386–396, 2022. [Online]. Available: https://doi.org/10.14710/j.gauss.11.3.386-396
4. Bank Mandiri, "Bank Mandiri profile: Livin' by Mandiri category," *Bank Mandiri*, [Online]. Available: https://www.bankmandiri.co.id. [Accessed: Oct. 6, 2024].
5. Z. I. Alfianti, "Sentiment analysis of cosmetic reviews on the Femaledaily website using Naive Bayes and Support Vector Machine based on Particle Swarm Optimization," Master's thesis, Computer Science Program, STMIK Nusa Mandiri, Jakarta, 2019.
6. W. Astriningsih, "Multi-aspect identification and sentiment analysis on hotel reviews using Deep Learning," Master's thesis, Informatics Program, Universitas Islam Indonesia, Yogyakarta, 2023.
7. U. I. Larasati, M. A. Muslim, R. Arifudin, and Alamsyah, "Improving the accuracy of Support Vector Machine using Chi-Square statistic and TF-IDF for movie review sentiment analysis," *Sci. J. Inform.*, vol. 6, no. 1. pp. 138–149, 2019. [Online]. Available: http://journal.unnes.ac.id/nju/index.php/sji
8. G. H. A. Noer, "Implementation of Naïve Bayes algorithm and TF-IDF in sentiment analysis of review data (Case study: E-commerce application Shopee reviews on Google Playstore)," Bachelor's thesis, Informatics Program, UIN Syarif Hidayatullah, Jakarta, 2023.
9. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, 1988. [Online]. Available: https://doi.org/10.1016/0306-4573(88)90021-0
10. C. Cortes, V. Vapnik, and L. Saitta, "Support-vector networks," *Mach. Learn.*, vol. 20. no. 3, pp. 273–297, 1995.
11. S. K. Shevade, S. S. Keerthi, S. Bhattacharyya, and K. R. K. Murthy, "Improvements to the SMO algorithm for SVM regression," *IEEE Trans. Neural Netw.*, vol. 11. no. 5, pp. 1188–1193, 2000. [Online]. Available: https://doi.org/10.1109/72.870050
12. O. A. M. López, A. M. López, and J. Crossa, *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer Nature Switzerland, 2022. [Online]. Available: https://doi.org/10.1007/978-3-030-89010-0
13. S. R. Munthe, "Application of Kernel Support Vector Machine (SVM) method for sentiment classification of Shopee Food on Twitter," Bachelor's thesis, Statistics Program, Universitas Diponegoro, Semarang, 2024.
14. Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evolutionary Computation*, pp. 69–73, 1998. [Online]. Available: https://doi.org/10.1109/ICEC.1998.699146
15. R. Al Ghivary, N. Wulandari, N. Srikandi, and F. A. M. Nazilatul, "The role of data visualization in supporting population data analysis in Indonesia," *PENTAHELIX J. Public Admin.*, vol. 1. no. 1. pp. 57–62, 2023.
16. A. Jalil, A. Homaidi, and Z. Fatah, "Implementation of Support Vector Machine algorithm for classifying stunting status in toddlers," *G-Tech: J. Appl. Technol.*, vol. 8, no. 3, pp. 2070–2079, 2024. [Online]. Available: https://doi.org/10.33379/gtech.v8i3.481
17. S. A. Puri, "Chronic kidney failure prediction using Support Vector Machine algorithm based on Particle Swarm Optimization (Case study: RS Roemani Muhammadiyah, Semarang)," Bachelor's thesis, Statistics Program, Universitas Diponegoro, Semarang, 2023.