APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

Sentiment Analysis to Measure Mobile Legend Application User Reviews Using Naive Bayes, SVM, Random Fores Algorithms, Decision Tree, dan Logistic Regression

Nevita Cahaya Ramadani, Master of Computer Science, Amikom Purwokerto University, cahayaramadaninevita@gmail.com

ABSTRACT

This study aims to analyze the sentiment of Mobile Legend application users on the Google Play Store using machine learning algorithms, including Naïve Bayes, Support Vector Machine (SVM), Random Forest, Decision Tree, and Logistic Regression. From the results of scraping data on 3989 reviews, it was found that Mobile Legend users in Indonesia tend to be neutral towards this application, with 1265 neutral comments, 1115 positive comments, and 1204 negative comments. Through model evaluation, SVM showed the highest accuracy of 87%, outperforming other algorithms such as Random Forest, Naïve Bayes, Decision Tree, and Logistic Regression. The conclusion of this study shows that SVM is an excellent choice for conducting sentiment analysis on Mobile Legend application user reviews. These results can be a guide for developers to understand user responses and improve app quality based on sentiment analysis findings. **Keywords** *: Analysis, Machine Learning, Netral*

1. INTRODUCTION

In today's digital era, mobile applications have become an integral part of the daily lives of many users. Mobile Legends, as one of the popular mobile games, has received great attention from users around the world. The success of an app can be measured not only by its technical performance, but also by the user experience expressed in the reviews. Sentiment analysis is a method that can provide deep insights into how users perceive and respond to an application. In this context, machine learning algorithms, such as Naive Bayes, SVM, Random Forest, Decision Tree, and Logistic Regression, are used to measure sentiment from Mobile Legends user reviews.

As in the previous study by Giovani et al (2020) entitled "Sentiment Analysis of Teacher Room Applications on Twitter". The purpose of this study is to determine the success of an application by conducting sentiment analysis of the application. It results that the best optimization application in this model is an SVM-based PSO algorithm with an accuracy value of 78.55% and an AUC of 0.853. This study succeeded in obtaining the most effective and best algorithm in classifying positive comments and negative comments related to the Ruang Guru

application. Further research by Adhi Putra (2021) entitled "Sentiment Analysis on User Reviews of Bibit and Bareksa Applications with KNN Algorithm". This study aims to analyze sentiment on user reviews of online investment applications, namely Biit and Bareksa. Based on the results obtained from the modeling stage using the nearest neighbors algorithm and a 60:40 ratio for training data and data testing, the accuracy and recall values generated from each application are 85.14%, 91.91%, and 76.44% for seedlings while for bareksa are 81.70%, 87.15%, 75.73%.

Further research by Wahyudi & Kusumawardana (2021) entitled "Sentiment Analysis on Grab Application Reviews on Google Play Store Using Support Vector Machine". The purpose of this study is to analyze the reviews of Grab application users on the Google Play Store, using sentiment analysis. This user review analysis uses the Support Vector Machine (SVM) method. Generating an analysis using the Support Vector Machine resulted in an accuracy of 85.54% and the most frequently reviewed positive review result is "ovo", while the most frequently reviewed negative review is "driver".

This research aims to apply various machine learning algorithms to measure user sentiment towards the Mobile Legends application. By analyzing user reviews, we can provide insight into how the app is received by the public. The results of the model evaluation will help in determining the effectiveness and reliability of each algorithm in classifying sentiment. The conclusion of this study is expected to provide useful insights for the development and maintenance of the Mobile Legends application.

2. METHODOLOGY

This study analyzes user reviews of the Mobile Legend application on Google Playstore using the Support Vectore Machine (SVM), Naïve Bayes, Random Forest, Decision Tree and Logistic Regression algorithms, with the research stages that can be seen in figure 1.

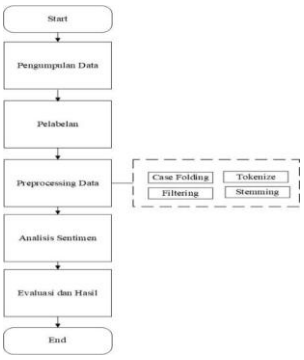


Figure 1. Research Methods

a. Data Collection

The data used is a review of the Mobile Legend application on the playstore. Data collection uses scraping techniques with the python library google play scraper. The data

taken was 10,000 reviews with the most relevant sorting. This data was taken on January 16, 2024.

b. Labeling

This labeling process is carried out so that the algorithm *Naïve Bayes, Support Vectore Machine, Random Forest, Decision Tree*and *Logistic Regression* can study datasets. The class consists of 3 which are 1 for Positive, -1 for Negative, and 0 for Neutral (Nugroho et al., 2021) .

c. Preprocessing Data

The data preprocessing process is an important step in sentiment analysis and text processing in general. The purpose of data preprocessing is to clean and prepare text data so that it can be processed properly by machine learning models. Some common steps in text data preprocessing are: (Suripto et al., 2022):

- *Casefolding*

  Namely removing urls from the content field, changing the text to lower *case*, removing mentions, removing hashtags, removing next characters, removing punctuation, removing *extra whitespace*.

- *Tokenize*

  It is a process of breaking down text or sentences into smaller units called "tokens." Tokens can be words, phrases, or other entities, such as numbers or symbols. The main purpose of tokenization is to make it easier to analyze and process text.

- *Filtering* (*Stopword Removal*)

  That is the stage of retrieving important words from the token results by using a *stoplist* algorithm (discarding less important words) or *wordlist* (storing important words).

  Stopwords are common words that usually appear in large numbers and are considered to have no meaning. Examples of *stopwords* in Indonesian are "who", "and", "in", "from", etc. The meaning behind the use of *stopwords* is that by removing words that have low information from a text, we can focus on important words instead.

- *Mood*

  It is a natural language processing technique that lowers the inflection of words to their root forms, thus aiding in the pre-processing of texts, words, and documents for the normalization of texts.

d. Sentiment Analysis

The sentiment analysis was carried out with 3 algorithms, namely *the Naïve Bayes algorithm, the Support Vectore Machine algorithm, Random Forest, Decision Tree,* and *Logistic Regression*. Here's an explanation of each algorithm:

- *Naïve Bayes*

  That is a classification method based on Bayes' probability theorem. It falls under the category of machine learning algorithms that use supervised learning methods. The algorithm is popular for text classification and document grouping tasks, but it can also be applied to a wide variety of data types (Widianto, 2019).

- *Supper Vectore Machine* (SVM)

ACSSE
APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

Namely machine learning algorithms used for classification and regression tasks. The main goal of an SVM is to build an optimal separator or hyperplane in a highdimensional space to separate instances from different classes (Samsudiney, 2019).

- *Random Forest*

  It is a machine learning algorithm used for classification, regression, and sorting tasks. This algorithm is a form of ensemble learning that combines several learning models (decision trees) to make more accurate and stable predictions (Trivusi, 2022).

After the labeling is done, the next feature extraction process is carried out by converting the text to a vector representation with TF-IDF (Term Frequency-Inverse Document Frequency). The formula for calculating TF-IDF, as follows (Delta Sierra, 2019):

$$IDFi = \log(\frac{D}{dfi})$$

Where D is the sum of all documents while dfi is the number of documents containing TF. The TF-IDF formula is as follows.

$$FIDF = TFxIDF$$

After that, the division of training data and test data was carried out by 80% and 20%. e.

Evaluation and Results

Performance evaluation is the process of measuring the extent to which a system or model succeeds or fails in achieving a specific goal. In the context of *machine learning*, performance evaluation is a way to measure how well a model can perform a particular task in a classification. To evaluate the performance of a model, a *confusion matrix* is used that contains original and predicted information. From the matrix, accuracy, precision, and recall can be known, as shown in table 1.

Tabel 1. Confusion Matrix

Class		**Predicted Class**		
		Positive	Negative	Neutral
Actual Class	Positive	True Positive (TP)	False Negative (FN)	False Netral (FNet)
	Negative	False Positive (FP)	True Negatif (TN)	False Netral (FNet)
	Neutral	False Positive (FP)	False Negative (FNet)	True Netral (TNet)

The explanation in table 1 can be seen as follows:

- *True Positive* (TP):
  The actual instance is in the Positive class and is correctly predicted as the Positive class by the model.
- *True Negatif* (TN):
  The actual instance is in the Negative class and is correctly predicted as a Negative class by the model.

- *True Netral* (TNet):
  The actual instance is in the Neutral class and is correctly predicted as the Neutral class by the model.
- *False Positive* (FP):
  The actual instance is in a Negative or Neutral class, but is incorrectly predicted as a Positive class by the model.
- *False Negative* (FN):
  The actual instance is in a Positive or Neutral class, but is incorrectly predicted as a Negative class by the model.
- *False Netral* (FNet):
  The actual instance is in a Positive or Negative class, but is incorrectly predicted as a Neutral class by the model.

From the confusion matrix can be used to calculate the values of accuracy, precision, recall and F1, here is the formula of accuracy.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision calculation to find out the comparison of the number of positive classes that are predicted correctly compared to the number of all positive classes. Here is umus precision.

$$precision = \frac{TP}{+ FP} TP$$

The recall calculation was carried out to find out the comparison of the number of positive classes that were predicted to be correct compared to the classes that were correctly positive. The following is the recall formula.

$$recall = \frac{TP}{TP + FN}$$

The calculation of *F-Measure* is obtained from *precision* and *recall*. The following is the formula *F1*.

$$F1 = \frac{2 x precision x recall}{precision + recall}$$

3. RESULTS AND ANALYSIS

3.1. Data Collection

This data collection is carried out by *scrapping python google play scrapper*, as shown in figure 2.

APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

	content	score	Year	Month	Day
8206	Untuk penyimpanan semakin update semakin membe...	1	2023	8	8
6717	Parahhhh...makin kesini makin parah pengaturan...	1	2023	8	8
6720	Saya pemain lama emel, makin lama emel sering ...	3	2023	8	9
6616	Tolong kembalikan performa Mobile Legends sepe...	4	2023	8	9
6615	Tolong diperbaiki sinyal dan kapasitas penyimp...	5	2023	8	10
...	...	...	...	...	...
8207	Fitur 'highlight' setelah update menghilang. D...	4	2024	1	14
1300	Ram 6 bisa lag mulu..... Padahal penyimpanan...	1	2024	1	14
8672	Game bagus tapi bosan Mau nanya Tujuan ngasih ...	1	2024	1	14
8482	Ehh min klu ngasih tim it yg waras dikit lah b...	1	2024	1	15
8995	Aqw kasik ulasan Bintang 1 aja . Game tambah l...	1	2024	1	15

10000 rows × 5 columns

Figure 2. Data Collection

To make it easier to see the *score value* , score visualization is carried out, as shown in figure 3.

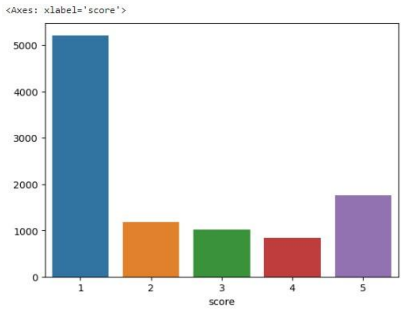


Figure 3. Score Visualization

In Figure 3, it can be seen *that the score* with a value of 1 is more than the others.

3.2. Labeling

In the labeling process, add a sentiment column with the criteria *of score* 1–2 to be negative with code -1, *score* 3 is neutral with code 0 and *score* 4–5 is positive with code 1, as shown in figure 4.

	content	score	Year	Month	Day	sentiment
8206	Untuk penyimpanan semakin update semakin membe...	1	2023	8	8	-1
6717	Parahhhh...makin kesini makin parah pengaturan...	1	2023	8	8	-1
6720	Saya pemain lama emel, makin lama emel sering ...	3	2023	8	9	0
6616	Tolong kembalikan performa Mobile Legends sepe...	4	2023	8	9	1
6615	Tolong diperbaiki sinyal dan kapasitas penyimp...	5	2023	8	10	1

Figure 4. Labeling

To make it easier to see the score labeling , a visualization of score labeling is carried out, as shown in figure 5.

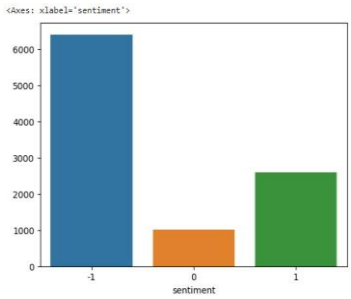


Figure 5. Labeling Visualization

DOI: https://doi.org/10.63254/acsse.v1i1.11

In Figure 5, it can be seen *that the score* with a value of -1 is more than the others.

3.3. Preprocessing

Data preprocessing is used to clean and prepare text data so that it can be processed properly by *machine learning models*.

- *Case Folding*

  Namely removing the url from the content column, changing the text to *lower case*, removing the mention, removing the hashtag, removing the next character, removing the punctuation, removing *the extra whitespace*, as shown in table 2.

Tabel 2. *Case Folding*

Dataset	It's strange even though the storage still has a lot of lag periods, a little immediately reloged, my gameplay is broken and made mooton, if I'm good at this, I'm not good at mooton, what's wrong with this winrate, the gameplay period also wants to be ruined, you can get a team that is not compact continuously, you can get opponents regardless of the skills, if it's like this, when every player has performance progress if it's accompanied by selfish teams, I'm not a pro player, beyond my thinking if I can get a team that continues to
CaseFolding	It's weird even though the storage still has a lot of lag periods, a little bit of a relog, my gameplay is broken, made a mooton if I'm good at being alert, I'm not good at mooton, what's wrong with this guy, the winrate has run out, the gameplay period is over, I want to be damaged, I can get a team that is not compact, continue to get opponents, I don't care if the skill is rich, continue to be like this, when every player has a performance progress, if it is together with a team, a selfish team, I am not a pro player, beyond my thinking If you can get that team, it will continue

- *Tokenize*

That is the process of breaking down text or sentences into smaller units called "tokens", as in table 3.

Tabel 3. *Tokenize*

Dataset	It's strange even though the storage still has a lot of lag periods, a little immediately reloged, my gameplay is broken and made mooton, if I'm good at this, I'm not good at mooton, what's wrong with this winrate, the gameplay period also wants to be ruined, you can get a team that is not compact continuously, you can get opponents regardless of the skills, if it's like this, when every player has performance progress if it's accompanied by selfish teams, I'm not a pro player, beyond my thinking if I can get a team that continues to
CaseFolding	It's weird even though the storage still has a lot of lag periods, a little bit of a relog, my gameplay is broken, made a mooton if I'm good at being alert, I'm not good at mooton, what's wrong with this guy, the winrate has run out, the gameplay period is over, I want to be damaged, I can get a team that is not compact, continue to get opponents, I don't care if the skill is rich, continue to be like this, when every player has a performance progress, if it is together with a team, a selfish team, I am not a pro player, beyond my thinking If you can get that team, it will continue

ACSSE
APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

Tokenizer	['weird', 'save', 'lag', 'live', 'relog', 'gameplay', 'me', 'broken', 'mooton', 'jago', 'sigapapa', 'jago', 'mooton', 'si', 'winrate', 'udah', 'out', 'exhausted', 'gameplay', 'damaged', 'team', 'compact', 'opponent', 'skill', 'his', 'rich', 'gini', 'player', 'progress', 'performance', 'together', 'team', 'team', 'selfish', 'pro', 'player', 'outside', 'thought', 'kalo', 'team']

- *Filtering*

That is, throwing away less important words or *wordlists* storing important words, as in table 4.

Table 4. *Filtering*

Dataset	It's strange even though the storage still has a lot of lag periods, a little immediately reloged, my gameplay is broken and made mooton, if I'm good at this, I'm not good at mooton, what's wrong with this winrate, the gameplay period also wants to be ruined, you can get a team that is not compact continuously, you can get opponents regardless of the skills, if it's like this, when every player has performance progress if it's accompanied by selfish teams, I'm not a pro player, beyond my thinking if I can get a team that continues to
CaseFolding	It's weird even though the storage still has a lot of lag periods, a little bit of a relog, my gameplay is broken, made a mooton if I'm good at being alert, I'm not good at mooton, what's wrong with this guy, the winrate has run out, the gameplay period is over, I want to be damaged, I can get a team that is not compact, continue to get opponents, I don't care if the skill is rich, continue to be like this, when every player has a performance progress, if it is together with a team, a selfish team, I am not a pro player, beyond my thinking If you can get that team, it will continue
Tokenizer	['weird', 'save', 'lag', 'live', 'relog', 'gameplay', 'me', 'broken', 'mooton', 'jago', 'sigapapa', 'jago', 'mooton', 'si', 'winrate', 'udah', 'out', 'exhausted', 'gameplay',
	'damaged', 'team', 'compact', 'opponent', 'skill', 'his', 'rich', 'gini', 'player', 'progress', 'performance', 'together', 'team', 'team', 'selfish', 'pro', 'player', 'outside', 'thought', 'kalo', 'team']
Filtering	['weird', 'save', 'lag', 'live', 'relog', 'gameplay', 'ku', 'broken', 'mooton', 'jago', 'sigapapa', 'jago', 'mooton', 'si', 'winrate', 'udah', 'out', 'out', 'out', 'gameplay', 'broken', 'team', 'compact', 'opponent', 'skill', 'his', 'rich', 'gin', 'player', 'advanced', 'performance', 'together', 'team', 'team', 'selfish', 'pro', 'player', 'outside', 'think', 'if', 'team']

- *Mood*

It is a natural language processing technique that lowers the inflection of words to their root forms, thus aiding in the pre-processing of texts, words, and documents for text normalization, as shown in table 5.

Table 5. *Mood*

Dataset	It's strange even though the storage still has a lot of lag periods, a little immediately reloged, my gameplay is broken and made mooton, if I'm good at this, I'm not good at mooton, what's wrong with this winrate, the gameplay period also wants to be ruined, you can get a team that is not compact continuously, you can get opponents regardless of the skills, if it's like this, when every player has performance progress if it's accompanied by selfish teams, I'm not a pro player, beyond my thinking if I can get a team that continues to

ACSSE
APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

CaseFolding	It's weird even though the storage still has a lot of lag periods, a little bit of a relog, my gameplay is broken, made a mooton if I'm good at being alert, I'm not good at mooton, what's wrong with this guy, the winrate has run out, the gameplay period is over, I want to be damaged, I can get a team that is not compact, continue to get opponents, I don't care if the skill is rich, continue to be like this, when every player has a performance progress, if it is together with a team, a selfish team, I am not a pro player, beyond my thinking If you can get that team, it will continue
Tokenizer	['weird', 'save', 'lag', 'live', 'relog', 'gameplay', 'me', 'broken', 'mooton', 'jago', 'sigapapa', 'jago', 'mooton', 'si', 'winrate', 'udah', 'out', 'exhausted', 'gameplay', 'damaged', 'team', 'compact', 'opponent', 'skill', 'his', 'rich', 'gini', 'player', 'progress', 'performance', 'together', 'team', 'team', 'selfish', 'pro', 'player', 'outside', 'thought', 'kalo', 'team']
Filtering	['weird', 'save', 'lag', 'live', 'relog', 'gameplay', 'ku', 'broken', 'mooton', 'jago', 'sigapapa', 'jago', 'mooton', 'si', 'winrate', 'udah', 'out', 'out', 'out', 'gameplay', 'broken', 'team', 'compact', 'opponent', 'skill', 'his', 'rich', 'gin', 'player', 'advanced', 'performance', 'together', 'team', 'team', 'selfish', 'pro', 'player', 'outside', 'think', 'if', 'team']
Mood	strange save directly relog broken gameplay mooton jago sigapapa jago mooton winrate has run out of broken gameplay compact opponent skills like players advanced performance along with selfish players unexpected

3.4.Sentiment Analysis

The sentiment analysis was carried out with 5 algorithms, namely *the Naïve Bayes algorithm, the Support Vectore Machine algorithm, Random Forest, Decision Tree,* and *Logistic Regression.* Before doing the analysis, first look at the positive, negative, and neutral reviews with *wordcloud*, as shown in figures 6,7,8.

Figure 6. Wordcloud Positive

Figure 7. Wordcloud Negatives

DOI: https://doi.org/10.63254/acsse.v1i1.11

ACSSE
APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

Figure 8. Wordcloud Neutral

After seeing the positive, negative and neutral wordclouds. Next, change the value to TF-IDF. Term frequency-inverse document frequency is a text vectorizer that converts text into usable vectors. It combines 2 concepts, Term Frequency (TF) and Document Frequency (DF). Furthermore, in this sentiment analysis, the stage is to split the data into training data and testing data by 80% and 20%, as shown in table 6.

Table 6. Split Data

	Training Data	**Data Testing**	**Total**
Amount of Data	15952	3989	19941

3.5.Evaluation and Results

Performance evaluation is the process of measuring the extent to which a system or model succeeds or fails in achieving a specific goal. The evaluation of the results can be seen in the following figure:

a. *Random Forest*

Before calculating *accuracy*, first calculate *the confusion matrix, as shown in figure 9.*

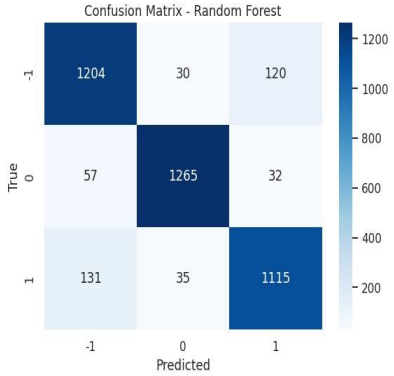


Figure 9. CM *Random Forest*

From the data results in figure 9, data with a total of 3989 user reviews obtained 1115 reviews were declared positive, 1204 comments were declared negative, and 1265 comments were declared neutral. Furthermore, the *classification report calculation is carried out* which includes *Precision score, recall score, f1-score* and *accuracy* value, as shown in figure 10.

APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

Classification Report:

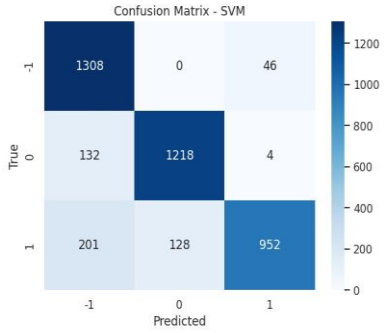
	precision	recall	f1-score	support
-1	0.76	0.96	0.85	1294
0	0.92	0.90	0.91	1289
1	0.94	0.72	0.81	1256
accuracy			0.86	3839
macro avg	0.87	0.86	0.86	3839
weighted avg	0.87	0.86	0.86	3839

Gambar 10. CR *Random Forest*

Figure 10 produces an accuracy value of 86% with a computing time of 28.98 seconds consisting of a training time of 28.78 seconds and a prediction time of 0.20 seconds.

b. *Support Vector Machine*

Before calculating *accuracy*, first calculate the *confusion matrix*, as shown in figure 11.



Gambar 11. CM *Support Vector Machine*

From the results of the data in figure 11, data with a total of 3989 user reviews obtained 952 reviews were declared positive, 1308 comments were declared negative, and 1218 comments were declared neutral. Next, the classification report calculation is carried out which includes *Precision score, recall score, f1-score* and *accuracy* value, as shown in figure 12.

Classification Report:

	precision	recall	f1-score	support
-1	0.80	0.97	0.87	1354
0	0.90	0.90	0.90	1354
1	0.95	0.74	0.83	1281
accuracy			0.87	3989
macro avg	0.88	0.87	0.87	3989
weighted avg	0.88	0.87	0.87	3989

Gambar 12. CR *Support Vector Machine*

Figure 12 produces an *accuracy value* of 87% with a computing time of 60.08 seconds consisting of a training time of 48.62 seconds and a prediction time of 11.46 seconds. c. *Naïve Bayes*

Before calculating *accuracy*, first calculate the *confusion matrix*, as shown in figure 13.

APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

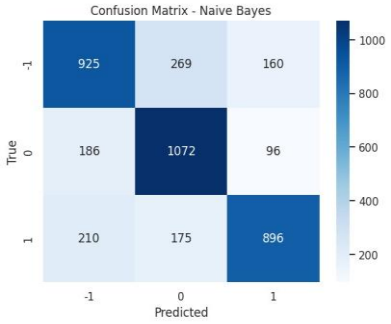


Figure 13. *CM Naïve Bayes*

From the results of the data in figure 12, data with a total of 3989 user reviews obtained 896 reviews were declared positive, 925 comments were declared negative, and 1072 comments were declared neutral. Next, the *classification report calculation is carried out* which includes *Precision score, recall score, f1-score* and *accuracy* value, as shown in figure 13.

Classification Report:

	precision	recall	f1-score	support
-1	0.70	0.68	0.69	1354
0	0.71	0.79	0.75	1354
1	0.78	0.70	0.74	1281
accuracy			0.73	3989
macro avg	0.73	0.72	0.73	3989
weighted avg	0.73	0.73	0.72	3989

Gambar 13. CR *Naïve Bayes*

In Figure 13, an *accuracy value* of 73% is produced with a computing time of 0.01 seconds consisting of a training time of 0.01 seconds and a prediction time of 0.0 seconds. d. *Decision Tree*

Before calculating *accuracy*, first calculate *the confusion matrix*, as shown in figure 14.

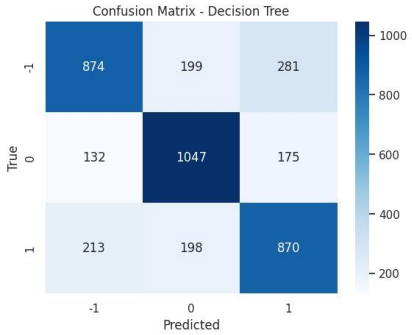


Figure 14. CM *Decision Tree*

From the data results in figure 14, data with a total of 3989 user reviews obtained 870 reviews were declared positive, 874 comments were declared negative, and 1047 comments were declared neutral. Furthermore, the calculation of *the classification report* is carried out which includes *Precision score, recall score, f1-score* and *accuracy* value, as shown in figure

15.

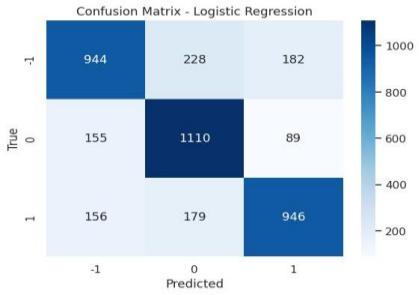
Classification Report:

	precision	recall	f1-score	support
-1	0.72	0.65	0.69	1354
0	0.73	0.77	0.75	1354
1	0.65	0.68	0.67	1281
accuracy			0.70	3989
macro avg	0.70	0.70	0.70	3989
weighted avg	0.70	0.70	0.70	3989

Gambar 15. CR *Decision Tree*

In Figure 15, an accuracy value of 70% is produced with a computing time of 4.39 seconds consisting of a training time of 4.39 seconds and a prediction time of 0.0 seconds. e. *Logistic Regression*

Before calculating *accuracy*, first calculate *the confusion matrix*, as shown in figure 16.



Gambar 16. CM *Logistic Regression*

From the data results in figure 16, data with a total of 3989 user reviews obtained 946 reviews were declared positive, 944 comments were declared negative, and 1110 comments were neutral. Furthermore, the calculation of *the classification Report* is carried out which includes *Precision score, recall score, f1-score* and *accuracy* value, as shown in figure 17.

Classification Report:

	precision	recall	f1-score	support
-1	0.75	0.70	0.72	1354
0	0.73	0.82	0.77	1354
1	0.78	0.74	0.76	1281
accuracy			0.75	3989
macro avg	0.75	0.75	0.75	3989
weighted avg	0.75	0.75	0.75	3989

Gambar 15. CR *Decision Tree*

Figure 15 produces an *accuracy value* of 75% with a computing time of 4.51 seconds consisting of a training time of 4.51 seconds and a prediction time of 0.0 seconds.

4. CONCLUSION

Based on user review data obtained from the results of scraping the Google Playstore Mobile Legend application, the tendency of the reviews submitted contains neutral comments. Mobile Legend users are neutral towards this application. Of the 3989 test data, there were 1265 comments with neutral sentiment, then 1115 comments with positive sentiment, and 1204 comments with negative sentiment. Here it can be concluded that Mobile Legend users in

Indonesia are neutral towards the application. Based on the results of the accuracy of the *Support Vectore Machine* algorithm of 87%, higher than the accuracy of the *Random Forest, Naïve Bayes, Decision Tree* and Logistic *Regression* Algorithms, it can be concluded that the *Support Vector Machine* algorithm is very good for analyzing the sentiment of Mobile Lege application user reviews. The suggestion of this study is to try to use other algorithm comparisons to produce better accuracy performance in analyzing user review sentiment such as KNN, BERT (*Bidirectional Encoder Representations from Transformers*) and LSTM (*Long Short Term Memory*) algorithms.

REFERENCES

Adhi Putra, A. D. Sentiment Analysis on User Reviews of Bibit and Bareksa Applications with KNN Algorithm. *JATISI (Journal of Informatics and Information Systems Engineering)*.2021.*8*(2), 636–646.

Delta Sierra. Algoritma TF — IDF. In *The Medium Blog* (p. https://dltsierra.medium.com/algoritma-tf-idf-633e).

Giovani, A. P., Ardiansyah, A., Haryanti, T., Kurniawati, L., & Gata, W. Sentiment Analysis of Teacher Room Applications on Twitter Using Classification Algorithm. *Journal of Technoinfo*. 2020. *14*(2), 115.

Nugroho, G., Murdiansyah, D., & Lhaksmana, K. Sentiment Analysis of the 2020 American Presidential Election on Twitter Using Naïve Bayes and the Support Vector Machine. *EProceeding of Engineering*. 2021. *8*(5), 10106–10115.

Samsudiney. (2019). A Simple Explanation of What Is SVM? | by Samsudiney | Medium. In *Https://Medium.Com/* (pp. 2–7). https://medium.com/@samsudiney/penjelasansederhana-tentang-apa-itu-svm-149fec72bd02

Suripto, Rahmanita, R. N., & Kirana, A. S. *Teknik Pre-processing dan Classification dalam Data Science – Master of Industrial Enginering*. https://mie.binus.ac.id/2022/08/26/teknik-pre-processing-dan-classification-dalam-datascience/

Trivusi. (2022). Random Algorithm Forest_ Its Definition and Uses. In *Trivusi* (p. 1). https://www.trivusi.web.id/2022/08/algoritma-random-forest.html

Wahyudi, R., & Kusumawardana, G. Sentiment Analysis on Grab Application on Google Play Store Using Support Vector Machine. *Journal of Informatics*. 2021. *8*(2), 200–207.

Widianto, M. H. (2019). Naive Bayes Algorithm | BINUS UNIVERSITY BANDUNG - Creative Technology Campus. In *Binus.Ac.Id*. https://binus.ac.id/bandung/2019/12/algoritma-naive-bayes/