Eduvest – Journal of Universal Studies
Volume 4 Number 05, May, 2024
p- ISSN 2775-3735- e-ISSN 2775-3727

IMPLEMENTATION OF K-NEAREST NEIGHBOR (KNN) AND LOGISTIC REGRESSION ALGORITHMS IN SENTIMENT ANALYSIS EVERMOS APP REVIEWS

Uswatun Hasana[1], Lulu Chaerani Munggaran[2]
[1,2] Magister Manajemen Sistem Informasi, Universitas Gunadarma, Indonesia
Email: uswatunhasana@student.gunadarma.ac.id

ABSTRACT

The Evermos application is one of the e-commerce applications for resellers and dropshippers in Indonesia. To find out the quality of the application requires an assessment from application users. One method that can be used is sentiment analysis. Sentiment analysis is one of the text mining techniques that can help in processing a judgment, complaint, perception or response to a particular object. In this study, the sentiment analysis carried out was the analysis of Evermos application review text on the Google Play Store accompanied by emoji conversion by comparing the K-Nearest Neighbor (KNN) model and Logistic Regression. The comparison of these two classification models was done to get the best accuracy in the sentiment analysis. The results of the two test models that have been carried out, the Logistic Regression model is a classification model that has the best accuracy results with a value of 98.1% in test data and 82.7% in training data.

KEYWORDS *Sentiment Analysis, Evermos, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression*

INTRODUCTION

The rapid development of technology today includes the increasingly popular process of online buying and selling, particularly in Indonesia. This is evidenced by a Global Web Index report indicating that Indonesia has the highest rate of e-commerce users in the world, with 90% of internet users aged 16-64 having made online purchases (Dirgantari et al., 2020). The increase in e-commerce users has been accelerated by the COVID-19 pandemic, which significantly boosted the technology sector, especially the growth of e-commerce users in Indonesia. The Indonesian E-commerce Association (idEA) reported that e-commerce growth in Indonesia rose by over 40% in 2021.

One impact of this growth in Indonesia is the emergence of various new e-commerce platforms, such as the Evermos app. Evermos is a social commerce reseller platform that sells various Muslim products in Indonesia. The app aims to help Indonesian SMEs grow profitably and support people who want to sell without stockpiling goods. Launched on the Google Play Store in 2018, it has been downloaded over 1 million times with more than 26,000 user reviews.

Every app has its strengths and weaknesses, which can be assessed through user satisfaction. User satisfaction with the Evermos app can be gauged from reviews on download platforms like the Google Play Store. User satisfaction is not only measured by user ratings but also through textual review comments submitted by users. To efficiently and effectively summarize information from thousands of user reviews, complaints, and responses, sentiment analysis is needed. Sentiment analysis helps companies understand the feedback and attitudes of a group or individual towards a particular topic by analyzing the overall context of documents (Setiayana, 2021).

Online reviews or feedback are not only in text form but can also include emojis. Emojis are graphic Unicode symbols that express a person's feelings. They are used to convey ideas or feedback that cannot be easily written in words . Emojis are highly correlated with the sentiment orientation of short texts (Zou & Xiang, 2022). In sentiment analysis, emojis can be processed during preprocessing by converting them into text (Azzahra & Wibowo, 2020).

There are three main approaches to sentiment analysis: machine learning-based, lexicon-based, and hybrid approaches (Thomas et al., 2021). One commonly used method in sentiment analysis research is the machine learning approach, where training data is labeled and used to train the model (Lestari et al., 2017). Examples of machine learning methods include K-Nearest Neighbor (KNN), Naive Bayes Classifier (NBC), Support Vector Machine (SVM), and Logistic Regression.

The performance of a sentiment analysis classification model using machine learning can be measured with a confusion matrix, which compares the model's predictions with actual data. This matrix can be used to calculate and evaluate Precision, Recall, F-score, and Accuracy (Budianita et al., 2022). Joseph Hartmann and colleagues highlighted that accuracy is a critical aspect of sentiment analysis methods (Thomas et al., 2021). Accuracy is the ratio of correct predictions to the total data (Hartmann et al., 2022).

Various previous studies have explored user review sentiment analysis for applications. For instance, Andry Novantika and Sugiman's study titled "Sentiment Analysis of Google Meet User Reviews Using SVM and Logistic Regression Methods" found that the SVM method with a linear kernel had better accuracy (87.02%) compared to Logistic Regression (85.17%) (Novantika & Sugiman, 2022). Another study by Muhammad Zaki Farhan, titled "Sentiment Analysis of Shopeefood Services on Twitter Using K-Nearest Neighbor, Support Vector Machine, and Decision Tree Methods," found that K-Nearest Neighbor (KNN) achieved better accuracy (71.49%) compared to SVM (70.63%) and Decision Tree (59.80%) (Farhan, 2023). However, this study did not implement emoji conversion in the preprocessing stage.

Based on the above, the researcher is interested in conducting a study comparing sentiment analysis methods for the Evermos app, incorporating emoji conversion. This study will compare the K-Nearest Neighbor (KNN) and Logistic Regression classification methods to determine which achieves the best accuracy in sentiment analysis of Evermos app reviews on the Google Play Store, including emoji conversion.

RESEARCH METHOD

This sentiment analysis research uses review data of the Evermos application sourced from the Google Play Store download platform. The data processed in this research comprises 13,500 Indonesian-language text reviews collected between June 2019 and May 2023. The study is implemented using the Python programming language.

The research process begins with collecting review data using web scraping, followed by data processing and analysis, including preprocessing, labeling, transformation, data mining, and evaluation. The final stage of this research is summarizing the results. The research framework is illustrated in Figure 1.

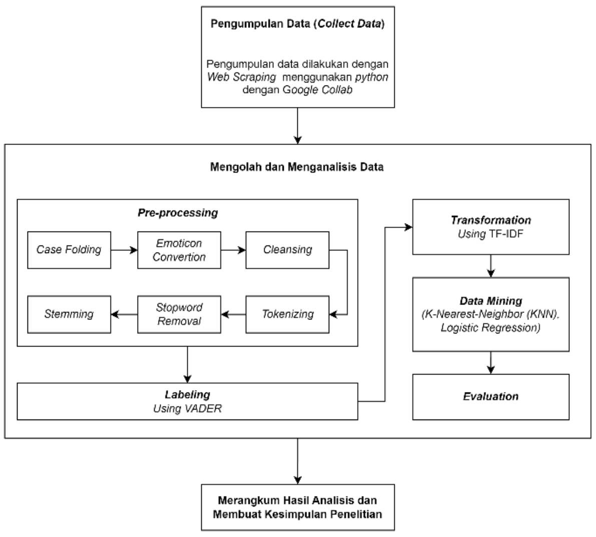

Pengumpulan Data (Collect Data)
Pengumpulan data dilakukan dengan Web Scraping menggunakan python dengan Google Collab
Mengolah dan Menganalisis Data
Pre-processing
Case Folding
Emoticon Convertion
Cleansing
Stemming
Stopword Removal
Tokenizing
Labeling Using VADER
Transformation Using TF-IDF
Data Mining (K-Nearest-Neighbor (KNN), Logistic Regression)
Evaluation
Merangkum Hasil Analisis dan Membuat Kesimpulan Penelitian

Figure 1. Research Framework

In Figure 1, the preprocessing process is limited to six steps: case folding, emoticon conversion, cleansing, tokenizing, stopword removal, and stemming. Next, sentiment labeling of the preprocessed text reviews is performed using the Valence Aware Dictionary and Sentiment Reasoner (VADER) method. The labeled data is then transformed into numerical data using the Term Frequency-Inverse Document Frequency (TF-IDF) method. This TF-IDF result is normalized using the Standard Scaler method, and the data is analyzed using the K-Nearest Neighbor (KNN) and Logistic Regression classification methods. The data mining process flow is illustrated in Figure 2.

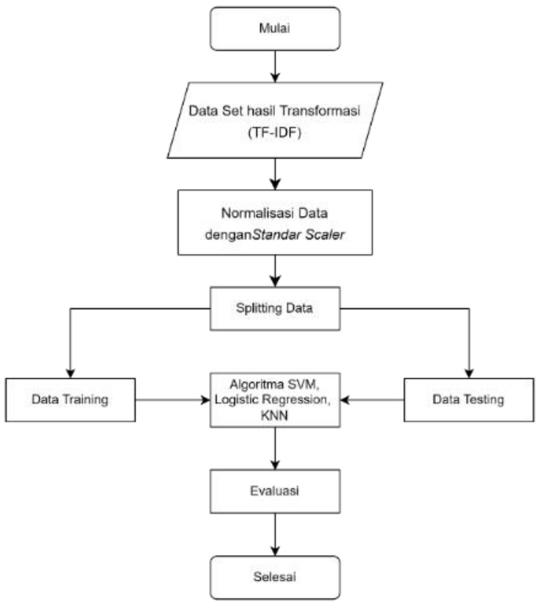

Mulai
Data Set hasil Transformasi (TF-IDF)
Normalisasi Data dengan*Standar Scaler*
Splitting Data
Data Training
Algoritma SVM, Logistic Regression, KNN
Data Testing
Evaluasi
Selesai

Figure 2. Data Mining Process Flow

The ratio used for splitting the data into training and test sets is 80:20. A larger amount of training data indicates that the data can represent the entire dataset with different characteristics (Wicaksono et al., 2021). The implementation results of the model in the training and testing data processes will be evaluated using a 3x3 confusion matrix. The evaluation will consider accuracy, execution time, and the total prediction error. The accuracy calculation based on the 3x3 confusion matrix can be seen in calculation (1):

$$Accuracy = \frac{TPos+TNet+TNeg}{TPos+FPosNeg+FPosNet+TNet+FNetPos+FNetNeg+TNeg+FNegNet+FNegPos}$$
(1)

RESULT AND DISCUSSION

Data Collection

Data collection is carried out by *web scraping* methods that are implemented using the python programming language by utilizing. *Google Play Scraper library*. Some of the results of review sentiment data obtained from the *web scraping* process can be seen in Table 1.

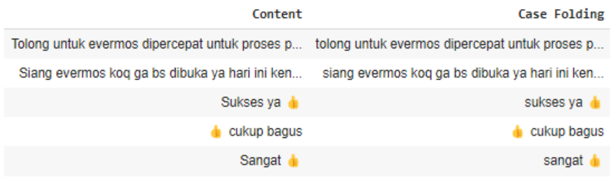
Table 1. Data from the *Scraping Process*

Content	**At**
3 stars first huh.. 😊 😊 You see: There is still a lot to improve. Never Order is not in accordance with the order .even though the ordered is clear and correct. But what is sent is different from Photos in the Catalog.	2023-05-19 07:02:25
Evermos It's great	2023-05-18 19:54:33

Good	2023-05-18 15:28:12
Semoga bermanfaat	2023-05-18 10:40:40
Recently I found Reseller application that is easy to use and apply	2023-05-179:16:44

Results of Data Processing and Analysis

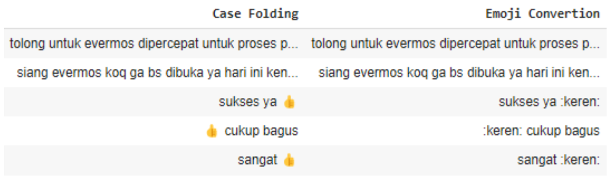
The next process after obtaining the object of research data is to carry out processing and analysis which begins with *preprocessing*. *Preprocessing* is carried out with six stages. The first stage is to do *casefolding* by changing the text to lowercase. Some examples of the results of the *casefolding* process can be seen in Figure 3.



Content	Case Folding
Tolong untuk evermos dipercepat untuk proses p...	tolong untuk evermos dipercepat untuk proses p...
Siang evermos koq ga bs dibuka ya hari ini ken...	siang evermos koq ga bs dibuka ya hari ini ken...
Sukses ya 👍	sukses ya 👍
👍 cukup bagus	👍 cukup bagus
Sangat 👍	sangat 👍

Figure 3. Preprocessing *Casefolding Results*

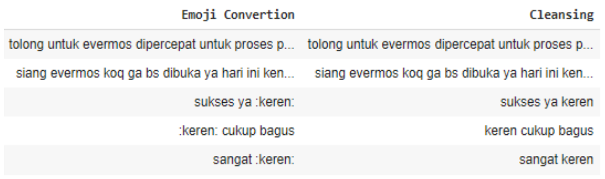
The second *preprocessing* stage is to perform emoji conversion on text containing emoji characters. At this stage the emoji is translated in the form of text describing the emoji. The results of the emoji conversion process can be seen in Figure 4.



Case Folding	Emoji Convertion
tolong untuk evermos dipercepat untuk proses p...	tolong untuk evermos dipercepat untuk proses p...
siang evermos koq ga bs dibuka ya hari ini ken...	siang evermos koq ga bs dibuka ya hari ini ken...
sukses ya 👍	sukses ya :keren:
👍 cukup bagus	:keren: cukup bagus
sangat 👍	sangat :keren:

Figure 4. Emoji Conversion *Preprocessing* Results

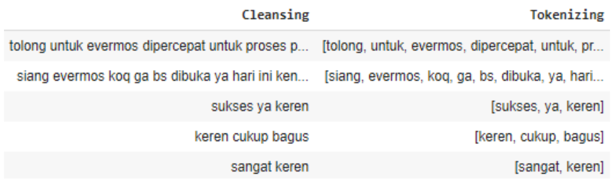
Text that has been processed for emoji conversion is then reprocessed by *cleansing data*. This process is used to remove attributes that are useless or have no effect on the analysis process. Figure 5 is an example of the output of the *cleansing process*.



Emoji Convertion	Cleansing
tolong untuk evermos dipercepat untuk proses p...	tolong untuk evermos dipercepat untuk proses p...
siang evermos koq ga bs dibuka ya hari ini ken...	siang evermos koq ga bs dibuka ya hari ini ken...
sukses ya :keren:	sukses ya keren
:keren: cukup bagus	keren cukup bagus
sangat :keren:	sangat keren

Figure 5. *Preprocessing Cleansing Results*

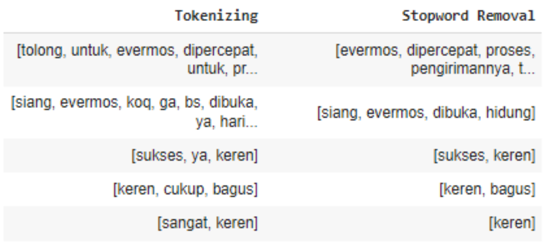
The next process after *cleansing* the data is *tokenizing*. At this stage the text is broken down into bundles of word tokens in the form of arrays to facilitate the word cleaning process. The results of the *tokenizing* process can be seen in Figure 6.



Cleansing	Tokenizing
tolong untuk evermos dipercepat untuk proses p...	[tolong, untuk, evermos, dipercepat, untuk, pr...
siang evermos koq ga bs dibuka ya hari ini ken...	[siang, evermos, koq, ga, bs, dibuka, ya, hari...
sukses ya keren	[sukses, ya, keren]
keren cukup bagus	[keren, cukup, bagus]
sangat keren	[sangat, keren]

Figure 6. Tokenizing *Preprocessing Results*

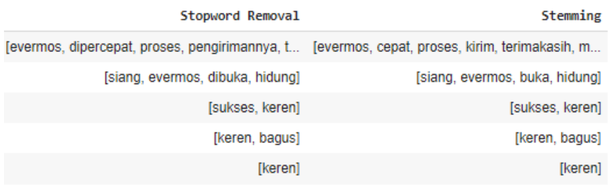
The results of the *tokenizing* process are then reprocessed by removing words that are not needed in the analysis process or called *the stopword removal process*. This process is carried out so that the analysis carried out is only words that can be useful in the analysis process. The results of the *stopword removal* process can be seen in Figure 7.



Tokenizing	Stopword Removal
[tolong, untuk, evermos, dipercepat, untuk, pr...	[evermos, dipercepat, proses, pengirimannya, t...
[siang, evermos, koq, ga, bs, dibuka, ya, hari...	[siang, evermos, dibuka, hidung]
[sukses, ya, keren]	[sukses, keren]
[keren, cukup, bagus]	[keren, bagus]
[sangat, keren]	[keren]

Figure 7. Tokenizing *Preprocessing Results*

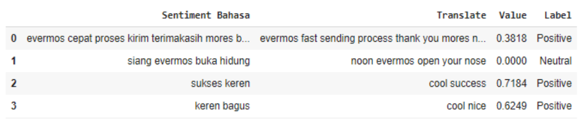
The final stage of *preprocessing* of this research is *stemming*. At this stage, the result of the *stopword removal* process will be the removal of word affixes so that the base word is obtained. This process is implemented by implementing *a literary library* that can remove Indonesian word affixes. The results of the *stemming* process can be seen in Figure 8.



Stopword Removal	Stemming
[evermos, dipercepat, proses, pengirimannya, t...	[evermos, cepat, proses, kirim, terimakasih, m...
[siang, evermos, dibuka, hidung]	[siang, evermos, buka, hidung]
[sukses, keren]	[sukses, keren]
[keren, bagus]	[keren, bagus]
[keren]	[keren]

Figure 8. Preprocessing Stemming *Results*

The preprocessing data is then reprocessed to be given sentiment labeling. The process of labeling text reviews is done automatically by implementing the VADDER *library*. Labels are divided into three categories, namely positive, neutral, and negative sentiment. Positive labels are marked with weights > 0, neutral labels are marked with weights = 0 and negative labels are marked with weights < 0. The results of the review labeling can be seen in Figure 9.



	Sentiment Bahasa	Translate	Value	Label
0	evermos cepat proses kirim terimakasih mores b...	evermos fast sending process thank you mores n...	0.3818	Positive
1	siang evermos buka hidung	noon evermos open your nose	0.0000	Neutral
2	sukses keren	cool success	0.7184	Positive
3	keren bagus	cool nice	0.6249	Positive

Figure 9. Sentiment Labelling *Results*

The labeling process of 13,500 Evermos application review datasets produced 9498 positive sentiment data, 3097 neutral sentiment data, and 905 negative sentiment data**.** The percentage result of the labeling process can be seen in Figure 10.

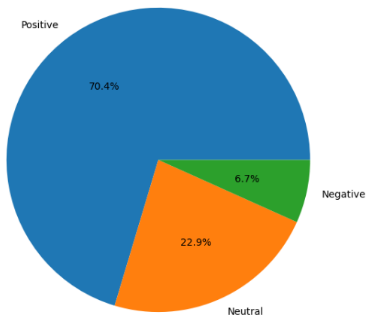
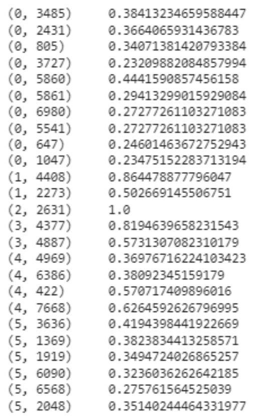


Figure 10. Sentiment Labelling *Results*

In the picture, it can be seen that this Evermos text review leads to a lot of positive sentiment. Data that has been labeled is processed into numerical or vector forms by applying TF-IDF weighting. This process is done to calculate how important a word is and how often it appears in the text. The TF-IDF process is implemented with *the skicit-learn python library* with the *TfidfVectorizer() function.* The results of the TF-IDF process can be seen in Figure 11 below.

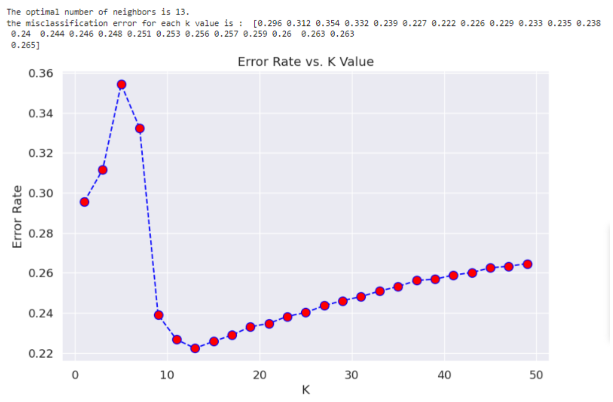


```
(0, 3485)     0.38413234659588447
(0, 2431)     0.3664065931436783
(0, 805)      0.34071381420793384
(0, 3727)     0.23209882084857994
(0, 5860)     0.4441590857456158
(0, 5861)     0.29413299015929084
(0, 6980)     0.27277261103271083
(0, 5541)     0.27277261103271083
(0, 647)      0.24601463672752943
(0, 1047)     0.23475152283713194
(1, 4408)     0.864478877796047
(1, 2273)     0.502669145506751
(2, 2631)     1.0
(3, 4377)     0.8194639658231543
(3, 4887)     0.5731307082310179
(4, 4969)     0.36976716224103423
(4, 6386)     0.38092345159179
(4, 422)      0.570717409896016
(4, 7668)     0.6264592626796995
(5, 3636)     0.4194398441922669
(5, 1369)     0.3823834413258571
(5, 1919)     0.3494724026865257
(5, 6090)     0.3236036262642185
(5, 6568)     0.275761564525039
(5, 2048)     0.35140244464331977
```

Figure 11. TF-IDF Process Results

The results of TF-IDF data are then processed to normalize data with *Standard Scaler* to normalize data weight to average 0 and standard deviation 1. The results of the data that have been normed are then divided into training data and test data with split validation fungi . The division result of 13500 data with a division ratio of 80:20 is 10800 training data and 2700 test data. The results of this data division are then analyzed with two classification models to be compared.

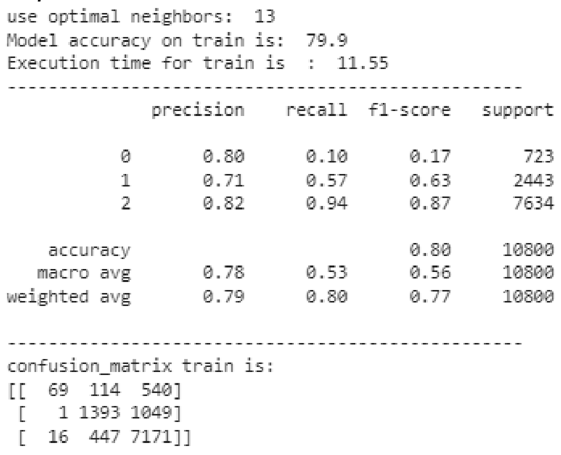
The first model implementation process is the *K-Nearest Neighbor* (KNN) algorithm. The first step to implementing this model is to find the optimal *value of neighbors*. In this study, the search for the optimal value of neighbors was carried out between values 1 - 50. The search results for the optimal value of *neighbors* can be seen in Figure 12.



```
The optimal number of neighbors is 13.
the misclassification error for each k value is :  [0.296 0.312 0.354 0.332 0.239 0.227 0.222 0.226 0.229 0.233 0.235 0.238
 0.24  0.244 0.246 0.248 0.251 0.253 0.256 0.257 0.259 0.26  0.263 0.263
 0.265]
```

Figure 12. Optimal Neighbors Search Results

In Figure 10 it can be seen that the optimal value of neighbors in this study is at k = 13. In k13, the *error rate* value is lower than the others so it is said to be the optimal neighbors value . The value *of neighbors* 13 is then implemented in training data testing and test data with the KNN model. The following are the results of the process of testing 10800 training data with the KNN algorithm model.can be seen in Figure 13 and *confusion matrix* visualization can be seen in Figure 14.



```
use optimal neighbors:  13
Model accuracy on train is:  79.9
Execution time for train is  :  11.55
------------------------------------------------
              precision    recall  f1-score   support

           0       0.80      0.10      0.17       723
           1       0.71      0.57      0.63      2443
           2       0.82      0.94      0.87      7634

    accuracy                           0.80     10800
   macro avg       0.78      0.53      0.56     10800
weighted avg       0.79      0.80      0.77     10800

------------------------------------------------
confusion_matrix train is:
[[  69  114  540]
 [   1 1393 1049]
 [  16  447 7171]]
```

Figure 13. KNN Test Results on Training Data

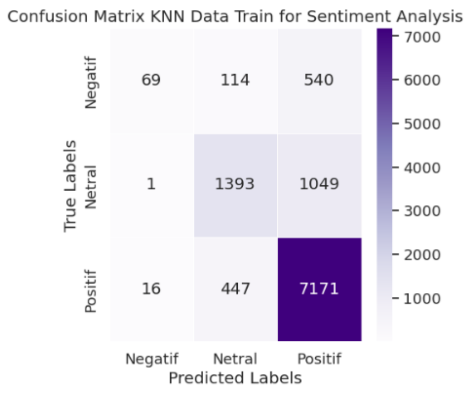


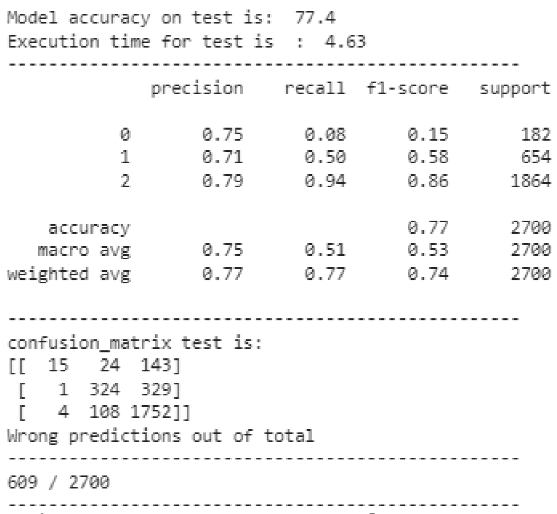
Figure 14. Confusion Matrix Visualization on KNN Training Data

In Figure 13 the performance evaluation of the KNN model on the test data produces a precision value for positive sentiment represented by number 2 is 82%, the precision value for neutral sentiment represented by number 1 is 71%, the precision value for negative sentiment represented by number 0 is 80%. The recall value for positive sentiment is 94%, neutral sentiment is 57%, and negative sentiment is 10%. The F1-Score for positive sentiment yields a value of 87%, neutral sentiment yields 63%, while negative sentiment yields 17%. In this evaluation, it can be concluded that the KNN model using training data has a good performance in classifying positive sentiments, but needs improvement in classifying negative and neutral sentiments.

The average accuracy value of all sentiments using the model on the training data based on Figure 13 is 79.9% or if rounded to 80%. The accuracy value is calculated based on the calculation (1) the following details of the calculation of the KNN accuracy value based on the *confusion matrix of* the training data:

$$Accuracy = \frac{7171+1393+69}{7171+16+447+1393+1049+1+60+114+540} \text{ x100\%}$$
$$Accuracy = \frac{8633}{10800} \text{ x100\% = 79,99\%}$$

The next test is the KNN model test using 2700 test data. The results of the testing process with the KNN algorithm model on the test data can be seen in Figure 15 and *the confusion matrix* visualization can be seen in Figure 16.



```
Model accuracy on test is:  77.4
Execution time for test is  :  4.63
--------------------------------------------------
              precision    recall  f1-score   support

           0       0.75      0.08      0.15       182
           1       0.71      0.50      0.58       654
           2       0.79      0.94      0.86      1864

    accuracy                           0.77      2700
   macro avg       0.75      0.51      0.53      2700
weighted avg       0.77      0.77      0.74      2700

--------------------------------------------------
confusion_matrix test is:
[[  15   24  143]
 [   1  324  329]
 [   4  108 1752]]
Wrong predictions out of total
--------------------------------------------------
609 / 2700
--------------------------------------------------
```

Figure 15. KNN Test Results on Test Data

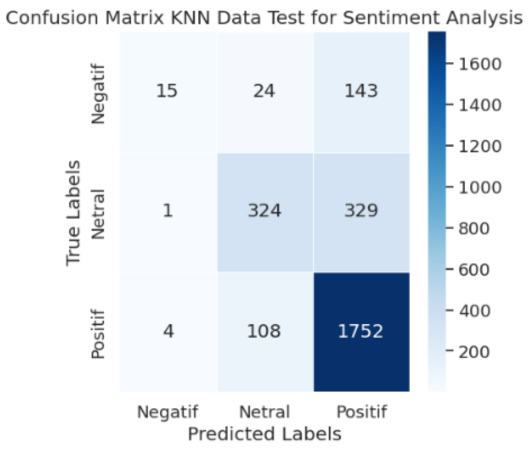


Figure 16. Confusion *Matrix Visualization* on KNN Test Data

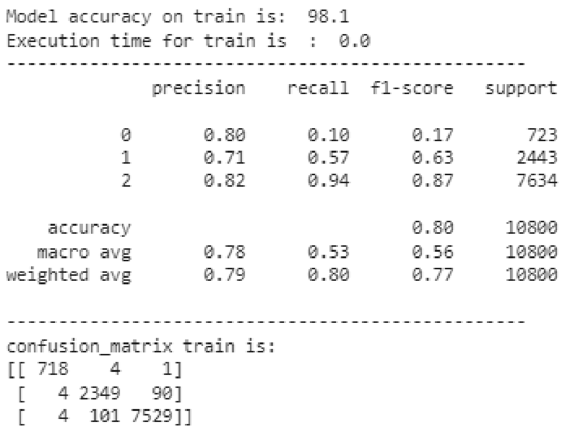
In Figure 15 it can be seen that the evaluation of the performance of the KNN model on the test data produces a precision value for positive sentiment represented by number 2 is 79%, the precision value for neutral sentiment represented by number 1 is 71%, the precision value for negative sentiment represented by number 0 is 75%. The recall value for positive sentiment is 94%, neutral sentiment is 50%, and negative sentiment is 8%. The F1-Score for positive sentiment yields a value of 86%, neutral sentiment yields 58%, while negative sentiment yields 15%. In this evaluation, it can be concluded that the KNN model using test data has good performance in the positive category, but does not have good performance in the neutral and negative label categories.

The average accuracy result value on Evermos application review sentiment test data using the KNN model based on Figure 15 was 77.4% with an execution time of 4.63 seconds with a total estimated prediction error of 609 data from 2700 test data. The results of the calculation of the accuracy of the KNN model based on *the confusion matrix* of test data:

$$Accuracy = \frac{1752+324+15}{1752+4+108+324+329+1+15+24+143} \text{x}100\%$$

$Accuracy = \frac{2091}{2700}$ x100% = 77,4%

The second implementation process is carried out with a *logistic regression* algorithm model using solver liblinear parameters and *auto multi_class*. The first stage carried out was to test 10800 training data. The test results can be seen in Figure 17 and the viasualization of the *confusion matrix* results of training data testing can be seen in Figure 18.



```
Model accuracy on train is:  98.1
Execution time for train is  :  0.0
---------------------------------------------------
              precision    recall  f1-score   support

           0       0.80      0.10      0.17       723
           1       0.71      0.57      0.63      2443
           2       0.82      0.94      0.87      7634

    accuracy                           0.80     10800
   macro avg       0.78      0.53      0.56     10800
weighted avg       0.79      0.80      0.77     10800

---------------------------------------------------
confusion_matrix train is:
[[ 718    4    1]
 [   4 2349   90]
 [   4  101 7529]]
```

Figure 17. Results of *Logistic Regression Testing* on Training Data

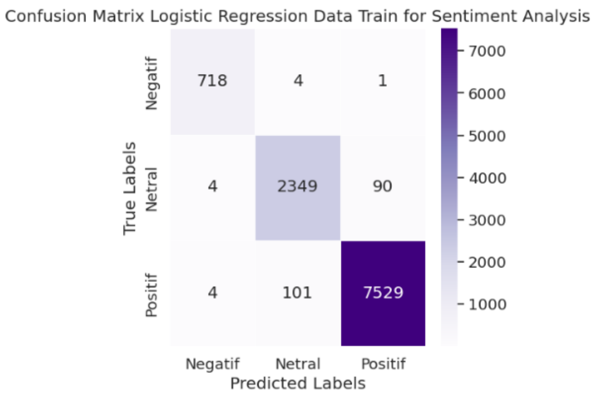


Figure 18. Confusion Matrix *Visualization* on Data Train *Logistic Regression*

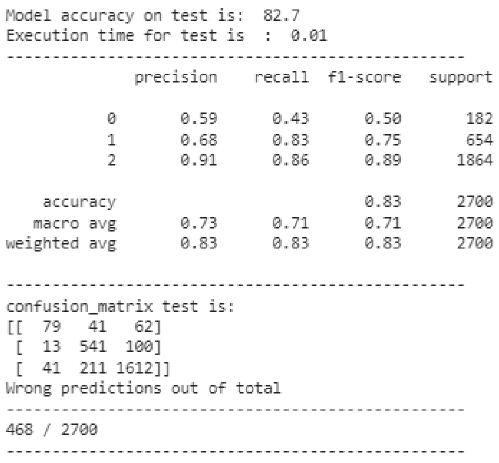
In Figure 17 it can be seen that the performance evaluation of the Logistic Regression model on the test data produces a precision value for positive sentiment represented by number 2 is 82%, the precision value for neutral sentiment represented by number 1 is 71%, the precision value for negative sentiment represented by number 0 is 80%. The recall value for positive sentiment is 94%, neutral sentiment is 57%, and negative sentiment is 10%. The F1-Score for positive sentiment yields a value of 87%, neutral sentiment yields 63%, while negative sentiment yields 17%. In this evaluation, it can be concluded that the Logistic Regression model using training data has a good performance in classifying positive sentiments, but needs improvement in classifying negative and neutral sentiments.

The average accuracy value for all sentiments in Figure 16 shows that the accuracy of testing training data using a *logistic regression* model is 98.1% with an execution time of 0.0 seconds. Details of the calculation of *logistic regression* accuracy values based on *the confusion matrix* of training data can be seen as follows:

$$Accuracy = \frac{7529+2349+718}{7529+4+101+2349+90+4+718+4+1} \text{x}100\%$$

$$Accuracy = \frac{10596}{10800} \text{x}100\% = 98{,}1\%$$

Further tests were carried out on 2700 test data. The results of testing test data with implementation odel *logistic regression* can be seen in Figure 19 and *visualization of confusion matrix* testing test data can be seen in Figure 20.



```
Model accuracy on test is:  82.7
Execution time for test is  :  0.01
--------------------------------------------------
              precision    recall  f1-score   support

           0       0.59      0.43      0.50       182
           1       0.68      0.83      0.75       654
           2       0.91      0.86      0.89      1864

    accuracy                           0.83      2700
   macro avg       0.73      0.71      0.71      2700
weighted avg       0.83      0.83      0.83      2700

--------------------------------------------------
confusion_matrix test is:
[[  79   41   62]
 [  13  541  100]
 [  41  211 1612]]
Wrong predictions out of total
--------------------------------------------------
468 / 2700
--------------------------------------------------
```

Figure 19. Logistic Regression Test Results on Test Data

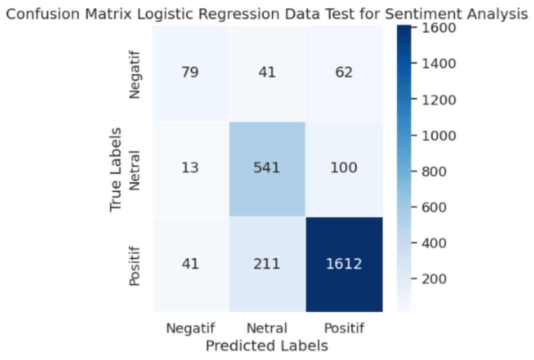


Figure 20. Confusion *Matrix Visualization on* Logistic Regression *Test Data*

In Figure 19 it can be seen that the performance evaluation of the Logistic Regression model on the test data produces a precision value for positive sentiment represented by number 2 is 91%, the precision value for neutral sentiment represented by number 1 is 68%, the precision value for negative sentiment represented by number 0 is 59%. The recall value for positive sentiment is 86%, neutral sentiment is 83%, and negative sentiment is 43%. The F1-Score for positive sentiment yields a value of 89%, neutral sentiment yields 75%, while negative sentiment yields 15%. In this evaluation, it can be concluded that the Logistic

Regression model using test data has good performance in the positive and neutral categories, but does not have good enough performance in the neutral and negative label categories.

The accuracy value of Evermos review sentiment test data using the *logistic regression model* in Figure 19 is 82.7% with an execution time of 0.01 seconds and with a total estimated prediction error of 468 data from 2700 test data. The following are the details of the calculation results of the accuracy of *the logistic regression* model based on *the confusion matrix* of test data:

$$Accuracy = \frac{1612+541+79}{1612+41+211+541+100+13+79+41+62} \text{ x100\%}$$

$$Accuracy = \frac{2232}{2700} \text{ x100\% = 82,7\%}$$

Comparison of accuracy results from the implementation process of two KNN classification models and Logistic Regression on training data and test data can be seen in Figure 21.

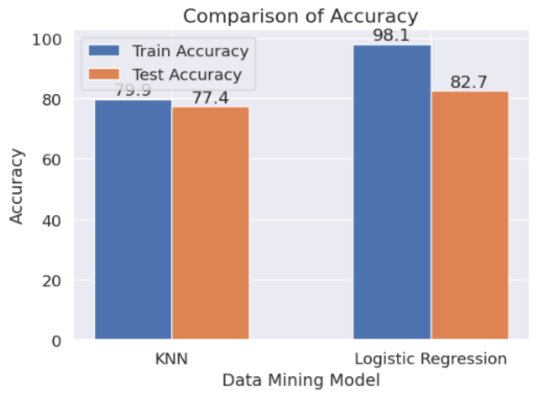


Figure 21. Visualization of Comparison of KNN Accuracy Results and *Logistic Regression*

In Figure 21, it can be concluded that the best accuracy value for the Evermos application review testing process on *the Google Play Store* is the Logistic Regression *model test* with accuracy results of 98.1% on the training data and 82.7% on the test data. The KNN model in this test is relatively lower than the *Logistic Regression* with an accuracy value of 79.9% training data and 77.4% test data so that the KNN model is not precise in this test process.

A comparison of execution time in the test process on the training data and the test data can be seen in Figure 22.

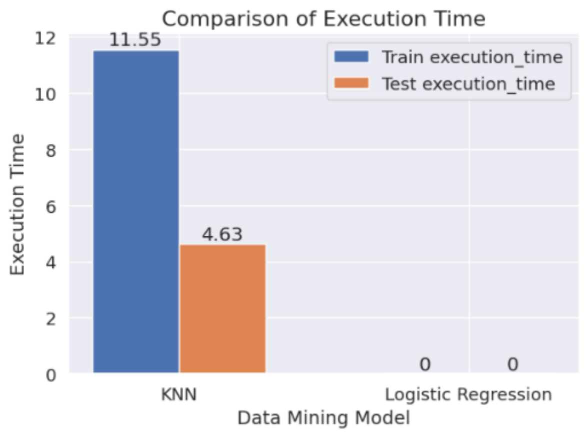


Figure 22. Visualization of KNN Testing Program Execution Time Comparison and *Logistic Regression*

In Figure 22, it can be concluded that the fastest program execution time for both training data and test data is testing *logistic regression models* with an execution time of 0 seconds in training data and test data. The KNN model is the model with the longest execution time with a time of 11.55 seconds for the training data and 4.63 seconds for the test data.

The next comparison is done by comparing the total prediction error in the test data. The total prediction error is calculated based on the difference between the amount of data predicted correctly and the total test data can be seen in Figure 23.

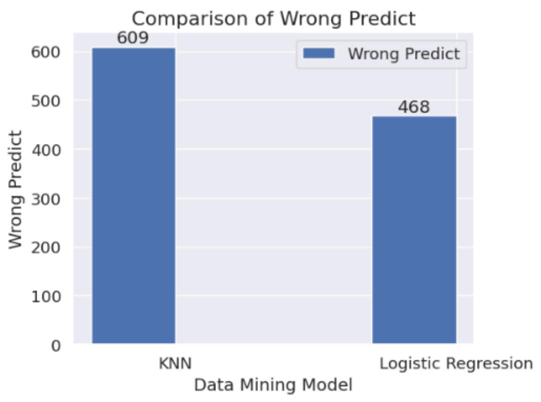


Figure 23. Visualization of Comparison of Total Prediction Error in Test Data Testing with KNN and *Logistic Regression models*

In Figure 23, it can be concluded that the *Logistic Regression* model is a model with a lower total prediction error compared to the KNN model. *The Logistic Regression* test has a total prediction error of 468 data while the KNN model has a prediction error with a total of 609 data.

CONCLUSION

Based on the tests that have been done, it can be concluded that the sentiment analysis process of Evermos application reviews in this study was successful. The review data trial with the Logistic Regression model resulted in an accuracy value of 98.1% on the training data with a 0-second execution process and 82.7% on the test data with a 0.01-second execution process. The total prediction error of the Logistic Regression model on the test data was 468. For the test process, the KNN model produced an accuracy value of 79.9% on the training data with an execution time of 11.5 seconds and 77.4% on the test data with an execution time of 4.63 seconds. Total prediction errors with the KNN model on the test data amounted to 609 data.

Based on the evaluation of the results that have been described, the best classification model for the sentiment analysis process of Evermos application reviews on the Google Play Store is the Logistic Regression classification model. This Logistic Regression model has a high accuracy value with fast program execution time compared to the KNN model, besides that the total error of prediction of test data with this model is relatively smaller than that of KNN. Based on these advantages, it can be concluded that the Logistic Regression model is the best classification model in conducting sentiment analysis of Evermos application reviews on the Google Play Store.

REFERENCES

Azzahra, S. A., & Wibowo, A. (2020). Analisis Sentimen Multi-Aspek Berbasis Konversi Ikon Emosi dengan Algoritme Naïve Bayes untuk Ulasan Wisata Kuliner Pada Web Tripadvisor. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *7*(4), 737. https://doi.org/10.25126/jtiik.2020731907

Budianita, E., Cynthia, E. P., Pranata, A., & Abimanyu, D. (2022). Pendekatan berbasis Machine Learning dan Leksikal Pada Analisis Sentimen. *Seminar Nasional Teknologi Informasi, Komunikasi Dan Industri (SNTIKI)*, 99–104.

Dirgantari, P. D., Hidayat, Y. M., Mahphoth, M. H., & Nugraheni, R. (2020). Level of use and satisfaction of e-commerce customers in covid-19 pandemic period: An information system success model (issm) approach. *Indonesian Journal of Science and Technology*, *5*(2), 261–270. https://doi.org/10.17509/ijost.v5i2.24617

Farhan, M. Z. (2023). ANALISIS SENTIMEN LAYANAN SHOPEEFOOD PADA TWITTER DENGAN METODE K-NEAREST NEIGHBOR , SUPPORT VECTOR MACHINE , DAN. *Jurnal Ilmiah Informatika*, *7*(2), 95–106.

Hartmann, J., Heitmann, M., Siebert, C., & Schamp, C. (2022). More than a Feeling: Accuracy and Application of Sentiment Analysis. *International Journal of Research in Marketing*, *40*, 75–87. https://doi.org/10.1016/j.ijresmar.2022.05.005

Lestari, A. R. T., Perdana, R. S., & Fauzi, M. A. (2017). Analisis Sentimen Tentang Opini Pilkada DKI 2017 Pada Dokumen Twitter Berbahasa Indonesia Menggunakan Naïve Bayes dan Pembobotan Emoji. *Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(12), 1718–1724.

Novantika, A., & Sugiman. (2022). Analisis Sentimen Ulasan Pengguna Aplikasi

Video Conference Google Meet menggunakan Metode SVM dan Logistic Regression. *PRISMA, Prosiding Seminar Nasional Matematika*, *5*, 808–813.

Setiayana, T. (2021). *Analisis Sentimen pada review aplikasi kesehatan HALODOC*.

Thomas, S., Yuliana, & Noviyanti. P. (2021). Study Analisis Metode Analisis Sentimen pada YouTube. *Journal of Information Technology*, *1*(1), 1–7. https://doi.org/10.46229/jifotech.v1i1.201

Wicaksono, A., Anita, A., & Padilah, T. N. (2021). Uji Performa Teknik Klasifikasi untuk Memprediksi Customer Churn. *Bianglala Informatika*, *9*(1), 37–45. https://doi.org/10.31294/bi.v9i1.9992

Zou, H., & Xiang, K. (2022). Sentiment Classification Method Based on Blending of Emoticons and Short Texts. *Entropy*, *24*(3). https://doi.org/10.3390/e24030398