Comparison Of Sentiment Classification Models At Sultan Hasanuddin Airport In Makassar

Muhammad Farhan Hermansyah[1], Dolly Indra[2*], Ramdaniah[3]

[1] *Department of Informatics Engineering, Universitas Muslim Indonesia, Jl. Urip Sumaharjo, Makassar, Indonesia. E-mail: mhdfrhank@gmail.com*

[2] *Department of Informatics System, Universitas Muslim Indonesia, Jl. Urip Sumaharjo, Makassar, Indonesia E-mail: dolly.indra@umi.ac.id*

[3] *Department of Informatics Engineering, Universitas Muslim Indonesia, Jl. Urip Sumaharjo, Makassar, Indonesia. E-mail: ramdaniah@umi.ac.id*

*Pemulis Korespondensi

Abstrak— **Analisis sentimen menggunakan machine learning penting untuk memahami persepsi publik terhadap layanan bandara. Renovasi Bandara Sultan Hasanuddin Makassar bertujuan meningkatkan kapasitas dan kenyamanan, namun tanggapan masyarakat terkait perubahan ini beragam. Penelitian ini membandingkan efektivitas tiga algoritma machine learning—Naive Bayes Multinomial, Support Vector Machine (SVM), dan Random Forest—dalam menganalisis sentimen ulasan pengguna terkait renovasi Bandara Sultan Hasanuddin di Makassar. Penelitian ini juga menerapkan teknik pemisahan data dan preprocessing teks menggunakan Google Colab dengan pemrograman berbasis Python, termasuk pembersihan data, stemming dengan Sastrawi, penghilangan stopword, dan ekstraksi fitur menggunakan metode TF-IDF dengan Unigram dan Bigram. Untuk mengatasi ketidakseimbangan kelas pada dataset, diterapkan teknik SMOTE. Data ulasan yang digunakan diambil dari Google Maps selama satu tahun terakhir. Hasil penelitian menunjukkan bahwa SVM dengan kernel linear memberikan performa terbaik dengan F1-score 92,3%, diikuti oleh Naive Bayes 83,7% dan Random Forest 81,9%. Unigram lebih efektif dibandingkan Bigram dalam ekstraksi fitur, dan SMOTE meningkatkan kinerja Naive Bayes pada dataset yang tidak seimbang, namun tidak berpengaruh signifikan pada SVM. Temuan ini memberikan rekomendasi untuk peningkatan layanan di Bandara Sultan Hasanuddin, seperti fasilitas kebersihan dan kenyamanan ruang tunggu.**

Kata Kunci — Naive Bayes; N-Gram; Random Forest; SMOTE; SVM

Abstract— **Sentiment analysis using machine learning is important to understand public perception of airport services. The renovation of Makassar's Sultan Hasanuddin Airport aims to increase capacity and comfort, but public responses to these changes are mixed. This study compares the effectiveness of three machine learning algorithms-Naive Bayes Multinomial, Support Vector Machine (SVM), and Random Forest-in analyzing the sentiment of user reviews related to the renovation of Sultan Hasanuddin Airport in Makassar. This research also applies data separation and text preprocessing techniques using Google Colab with Python-based programming, including data cleaning, stemming with Sastrawi, stopword removal, and feature extraction using TF-IDF method with Unigram and Bigram. To overcome class imbalance in the dataset, the SMOTE technique is applied. The review data used is taken from Google Maps for the past year. The results showed that SVM with linear kernel gave the best performance with F1-score 92.3%, followed by Naive Bayes 83.7% and Random Forest 81.9%. Unigram is more effective than Bigram in feature extraction, and SMOTE improves the performance of Naive Bayes on unbalanced datasets, but has no significant effect on SVM. The findings provide recommendations for service improvements at Sultan Hasanuddin Airport, such as cleaning facilities and waiting room comfort.**

Keywords — Naive Bayes; N-Gram; Random Forest; SMOTE; SVM

I. INTRODUCTION

Airports are one of the key elements in the air transportation system that connects various regions, both on a domestic and international scale. Apart from being a transit point for passengers and goods, airports have a strategic role in supporting the global economy [1]. With increasing population mobility and economic activity, the demand for air travel continues to increase every year. Renovating the development of airport facilities and services is an important step to overcome capacity limitations and improve service efficiency. One such effort can be seen in the massive renovation of Sultan Hasanuddin Airport in Makassar, South Sulawesi, which aims to increase terminal capacity, facility modernization, and passenger comfort [2]. However, this large-scale renovation effort elicited various responses from the public, including user reviews reflecting their experiences. The quality of services and facilities at

https://doi.org/10.35760/ik.2025.v30i1.14139

the airport is a crucial factor affecting the user experience, which is reflected in customer reviews of cleanliness, comfort, passenger flow efficiency, and staff attitude. A key challenge is how to comprehensively understand these reviews to evaluate the impact of renovations on user perceptions. How to interpret these reviews effectively, given that many reviews are ambiguous or simply describe facts without any clear feelings or sentiments. This is where sentiment analysis is important to evaluate the impact of renovations on users' perceptions of airport services and facilities, and provide insights for managers to improve service quality.

In the face of these challenges, machine learning technologies are an effective solution for classifying sentiment from user reviews. These techniques enable systematic and automated analysis of text data to identify positive or negative sentiment patterns. Naive Bayes, Support Vector Machine (SVM), and Random Forest methods are algorithms that are often used in sentiment analysis due to their good performance in processing text data [3], [4]. This approach allows public perceptions of the Sultan Hasanuddin Airport renovation to be mapped more clearly and objectively. Therefore, developing a model to understand the context and handle language variations used in user reviews is important. Most studies have also only compared two machine learning algorithms, limiting our understanding of the potential of different algorithms in more complex contexts.

Research on sentiment analysis of airports has grown rapidly in recent years, with many studies focusing on factors that influence passengers' perceptions of airport service quality. The first study highlighted the impact of extreme weather, especially lightning, which affects flight safety and airport operations. The ineffectiveness of weather warning systems often generates negative sentiment among passengers, while positive sentiment is formed when airports are responsive to extreme weather and improve safety procedures [5]. The second study addresses international connectivity, which is key in shaping sentiment. Although Beijing Airport has many international flights, the quality of connectivity, such as flight frequency and seat availability, is still low, causing negative sentiment. Increasing flight frequency and expanding international routes can improve passengers' positive sentiment [6]. The third study showed challenges in immigration control, with Airline Liaison Officers (ALOs) at Soekarno-Hatta and Ngurah Rai Airports. Unclear procedures and limited access can trigger negative sentiment from passengers, while more transparent and efficient procedures can generate positive sentiment [2]. In addition, comparing passenger perceptions at Sultan Hasanuddin Airport with other international airports such as Changi Airport in Singapore or Incheon Airport in South Korea can provide additional insights for service quality improvement. This comparative analysis can help identify best practices that can be adopted and adapted according to the local context. Overall, factors such as flight safety, international connectivity, and management of immigration control play a big role in shaping passenger sentiment towards airports. The use of technologies such as machine learning can help airports understand and improve the passenger experience more effectively.

This research aims to provide data-driven recommendations that can be used by the management of Sultan Hasanuddin Airport to improve service quality and improve areas of concern for users, such as cleanliness, waiting room comfort, and passenger flow efficiency. As such, this research contributes to the development of machine learning-based sentiment analysis methods and opens up opportunities to develop similar systems at other airports in Indonesia, which can improve user experience more broadly and sustainably.

II. RESEARCH METHOD

The method of this research consists of several stages, as shown in Figure 1. First, data is collected through a crawling process from the Google Maps platform. In the preprocessing stage, data cleaning and case folding, normalization, stemming, filtering, stopword removal, and tokenizing are performed [7]. This research also compares the use of unigram and bigram in feature extraction using TF-IDF method [8], [9]. In addition, comparisons were also made between data that was balanced using SMOTE and those that were not [10], [11]. Three classification models, Naive Bayes, SVM, and Random Forest, were applied to analyze sentiment [12], [13]. Afterwards, predictions were made on the test data, and an overall evaluation was conducted to determine the best model that was most effective in classifying public sentiment towards Sultan Hasanuddin Airport.

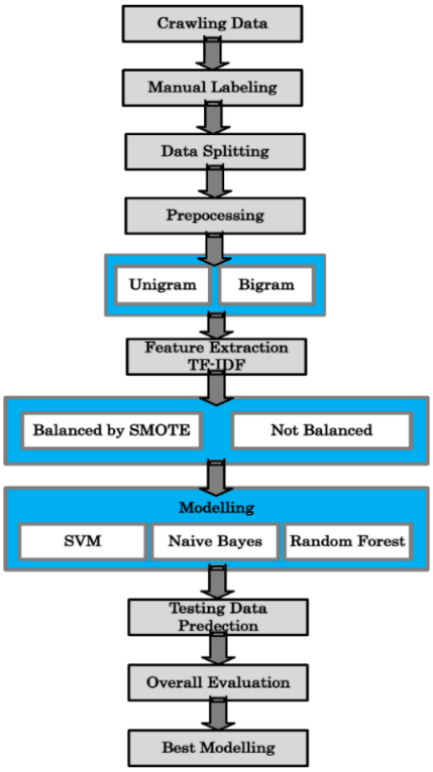


Figure 1. Research Method

Crawling Data

Data collection is the first step in this research, which involves scraping from the Google Maps platform. The review data regarding Makassar Sultan Hasanuddin Airport was collected manually by copying the reviews one by one using browser extensions such as Stop Smooth Scrolling. All data collected was copied into an Excel file to ensure that the data could be properly processed in the next stage. A total of 1,200 reviews were taken from reviews given in the last one year. To ensure the quality of the data, manual verification was conducted to remove duplicate or irrelevant reviews. In addition, Google Colab was used as a programming platform to perform further data processing. The collected data is then processed through a preprocessing stage to classify the sentiment.[14]

Manual Labelling

Manual labeling is the process of manually tagging data with specific categories or labels by researchers or domain experts. This step is often performed on unstructured or unlabeled datasets, such as text, images, or videos. Manual labeling is important for building datasets that can be used in training machine learning models, especially in classification tasks. Although time- and resource-consuming, manual labeling ensures the accuracy and relevance of the labels assigned to the data. Once the data is collected, labeling is done manually on each data Rating 1 is labeled as negative, while rating 5 is labeled as positive [15]. For ratings 2, 3, and 4, each review text is read in detail to determine whether the sentiment contained is positive or negative. In addition, ratings 1 and 5 are also reviewed to ensure more accurate results, as not all ratings 1 are always negative, and ratings 5 are not always positive [16].

Data Splitting

Data sharing separates the dataset into subsets for training, validation, and testing of machine learning models. Training data is used to teach the model to recognize patterns and relationships in the data. In contrast, validation data optimizes hyperparameters and prevents overfitting by assessing model performance during training. Test data, which is separate from the previous two subsets, is used to objectively evaluate the final performance of the model on data that it has never seen before. Proper data sharing ensures that the developed model can generalize well on new data, thus improving accuracy and reliability in real-world applications [17]. Next, the labeled data is divided into three parts: 70% Training Data, 20% Validation Data, and 10% Testing Data. This division is done to evaluate the performance of

the developed model, ensure the model works well, and avoid overfitting [18]. The data division is also done in stages to maintain the balance of positive and negative sentiment distribution in each data set.

Preprocessing Data

Data preprocessing is an important step in preparing data before it is used in machine learning models. In this research, the entire preprocessing process is done using Python and various related libraries in Google Colab, which provides convenience in data processing and text processing [19]. Some of the libraries used include Sastrawi for stemming, scikit-learn for TF-IDF feature extraction and data processing. The following are the stages carried out during the preprocessing process:

1. Cleaning & Case Folding*:*

The data is cleaned by removing irrelevant elements such as special characters, numbers, and punctuation marks. Case folding is done by converting the entire text to lowercase to reduce variations caused by different capitalizations. This process also includes removing unnecessary characters such as punctuation marks, extra spaces, emojis, numbers, stand-alone words such as "its", "anyway", and also removing endings from words. All text is converted into a consistent format by converting letters into lowercase using the NLTK library [20].

2. Normalization

Normalization aims to convert text into a uniform format. In this research, we implemented normalization by using three slangword dictionaries specific to this data, namely Slangword by Ramaprakoso, Slangword by pujangga, and Slangword by boy. Each of these slangwords is processed to be replaced with a more common or standard word. This includes the replacement of abbreviations and other nonstandard forms to more consistent forms. For example, numbers and abbreviations were converted into more common words to make further analysis easier. This process helps to reduce data variation and improve model accuracy [21].

3. Stemming

Stemming is performed using the Literature library to reduce words to their base or root form. For example, the words "running", "runner", and 'runs' are simplified to the base form "run". This helps to unify words with similar meanings, improve the efficiency of text analysis and increase the accuracy of the model [21].

4. Filtering & Stopword

Stopword removal is the process of removing words that do not have much meaning for analysis, such as conjunctions, prepositions, or articles (e.g. "and", 'or', "the", etc.). In this research, a manual stopword dictionary was created in-house, which contains words that are considered unimportant in the context of sentiment analysis of Sultan Hasanuddin Airport reviews. The words in this manual stopword dictionary are used to remove words that do not contribute important information, which helps to reduce the dimensionality of the data and focus on more meaningful words. This stopword dictionary was created specifically for this data to better fit the local context [22].

5. Tokenizing

Tokenization is the process of breaking down text into smaller units such as words or phrases. This process enables text analysis by separating each token so that it can be processed separately in later stages. The tokenization process is performed using the NLTK library [3].

6. N-Gram

N-Gram is a technique for breaking text into sequences of consecutive words, which is used in text analysis and language modeling to capture the context of adjoining words [19]. In this research, the analysis is done with two different approaches for comparison on the final results:

a. Unigram*:*

Unigrams are n-grams with n=1, meaning that each token is considered individually without regard to the context of the surrounding words. For example in the sentence "I eat rice", the unigrams are "I", 'eat', and "rice". Unigrams are often used in simpler models because of their simplicity, but they do not capture the relationship between words.

b. Bigram*:*

Bigrams are n-grams with n=2, which represent pairs of consecutive words. In the sentence "I eat rice", the bigrams are "I eat" and "eat rice". Bigrams are more effective in capturing context than

unigrams, which enhances the model's ability to understand the relationship between words and improves performance.

Feature Extraction TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a method to measure the importance of a word in a document relative to the set of documents. It is used to extract important features from text for use in machine learning. With TF-IDF, we can identify the most relevant and important words in the text, which will help the model focus more on meaningful words. The steps in calculating word weights are as follows [4]:

a. Calculating Term Frequency (TF) for each word: TF is calculated by breaking down the sentence into individual words, with each word assigned a value of 1.
b. Calculating Document Frequency (DF) for each word: DF is determined by summing the TF values for each word.
c. Calculating Inverse Document Frequency (IDF): After calculating TF, the next step is to compute the inverse DF.

The formula for IDF is shown in Formula 1.

$$IDF(w) = log\left(\frac{N}{DF(w)}\right) \quad (1)$$

The TF-IDF formula is shown in Formula 2:

TF-IDF is the product of TF and IDF:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t) \quad (2)$$

It is used to give higher weights to words that appear infrequently across documents, but frequently in a single document, as they are considered more relevant for classification [17].

Data Balancing

Data balancing overcomes class imbalance in datasets, where the majority class has more samples. Techniques such as oversampling, undersampling, and algorithms that are sensitive to class imbalance are used to balance the data. SMOTE (Synthetic Minority Oversampling Technique) adds data to the minority class to increase its representation, helping the model recognize the patterns of both classes more accurately. Experiments comparing model performance with SMOTE and unbalanced datasets can evaluate the effect of data balancing on model performance [10].

Modelling

Model building is the stage of applying machine learning algorithms or statistical methods to training data to create models that can recognize patterns and make predictions. This process involves selecting an algorithm that suits the type of data and research objective, such as regression, classification, or clustering. The model is then trained using the processed training data, optimizing the model parameters to minimize prediction errors. The result of this stage is a model that is ready to be tested and evaluated further using validation and testing data. The next stages are carried out in this study.

1. Parameter Setting

Parameter settings are used to optimize model performance by assigning specific values to each algorithm while exploring how different parameter configurations affect the analysis results. In SVM, a linear kernel is employed, where the C parameter controls regularization. Small values (0.01) result in a wider margin, while larger values (10.0) focus on more accurate classification [7]. In Naïve Bayes, the parameter used for Laplace smoothing prevents zero probabilities for infrequent words, with smaller values having minimal impact and larger values providing more aggressive smoothing. For Random Forest, exploration was carried out by testing various combinations of parameter values for each algorithm to understand their impact on model performance and identify the best configuration for optimal accuracy and generalization. The parameter settings are shown in Table 1.

Table 1. Parameter Setting

Algorithm	Distance / Function	Parameter	Value
Naïve Bayes	Multinomial	Alpha	0.1; 0.5; 1.0; 5.0; 10.0
SVM	Linear	C	0.01; 0.1; 1; 10; 100
Random Forest	Classifier	n-estimators	10; 50; 100; 200; 300

2. The Machine Learning algorithms used are

The Naive Bayes algorithm is a statistical classification method used to predict the probability of class membership. It relies on probability theory to determine the highest likelihood of classification by examining the frequency of each class in the training data. The basic formula of Naive Bayes is shown in Formula 3:

$$P(C \mid X) = \frac{P(X|C)\ x\ P(C)}{P(X)} \quad (3)$$

Description:

P(C | X) is the posterior probability that class *C* is assigned feature *X*, *P(X | C)* is the probability that feature *X* is assigned to class *C*, *P(C)* is the prior probability of class *C*, *P(X)* is the prior probability of feature *X* (often ignored because it is constant for all classes). Naive Bayes is very useful for probability-based classification, especially on large text data. Assuming that all features are independent, the model works fast and efficiently, even on datasets with many features. It is suitable for tasks such as text classification, spam filtering, and sentiment analysis, particularly with large datasets.

SVM is a machine learning method based on the principle of Structural Risk Minimization (SRM), which aims to find the best hyperplane that separates two classes in the input space. In simple terms, SVM seeks the optimal hyperplane, known as the "decision boundary," to distinguish between the two classes of data.

The SVM model is formulated in Formula 4:

$$w^T.x\ +\ b\ =\ 0 \quad (4)$$

Where *w* is the weight vector, *x* is the feature vector, *b* is bias.

The maximum margin is calculated by minimizing the loss function as shown in Formula 5 and Formula 6:

$$min\frac{1}{2}|w|^2 \quad (5)$$

With Condition:

$$y_i(w^T.x_i + b) \geq 1, \forall i \quad (6)$$

SVM is effective in finding the optimal hyperplane that separates classes with the maximum margin, making it particularly useful for high-dimensional data [30]. This model excels in handling data with complex and non-linear patterns through the use of kernels. SVM is especially suitable when the data has a large number of features relative to the number of samples.

Random Forest is a machine learning algorithm that falls under the ensemble learning category, which involves combining multiple predictive models to enhance prediction accuracy and stability. Developed by Leo Breiman and Adele Cutler, this algorithm has gained popularity for its robust ability to handle various types of data and problems, applicable to both classification and regression tasks [23]. The formula is shown in Formula 7:

$$\hat{c}\ = arg\ max\ \sum_{t=1}^{T} I\ (C_t = c) \quad (7)$$

Description:

$\hat{c}$ is the final class prediction by Random Forest, *C* is the set of all possible classes, *T* is the total number of trees in the Random Forest, *C_t* is the class prediction by the tttth tree, $I\ (C_t = c)$ is an indicator function that takes the value 1 if the tttth tree predicts the ccc class, and 0 otherwise. Random Forest is useful for handling data with class imbalance and provides stable results because it incorporates many decision trees. This model is excellent at capturing non-linear relationships and handling data that has missing values or less important features. Random Forest is suitable for variable data and is often used in prediction tasks that require high accuracy.

Testing Data Prediction

Test data prediction is the process where the trained model is applied to a separate subset of data to generate predictions. Since test data is not used during training, it provides an objective assessment of the model's performance on unseen data. This step is crucial for evaluating how well the model can generalize knowledge from the training data to new, real-world data. The prediction results are then compared with the actual values to measure the accuracy and effectiveness of the model, as well as its readiness for practical application.

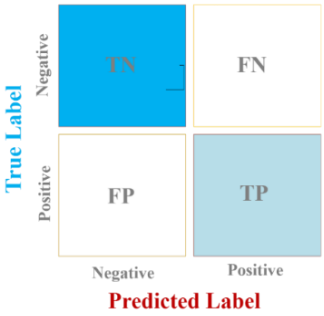


Figure 2. Confusion Matrix

Overall Evaluation

Overall evaluation is the stage where the model's performance is assessed using metrics such as accuracy, precision, recall, and F1-score based on the test results. Various metrics are used to evaluate the performance of the classification algorithm. Accuracy measures the percentage of correct predictions against the total data, precision assesses the proportion of correct positive predictions, recall gauges the model's ability to identify positive classes, and F1-score represents the harmonic average of precision and recall. This evaluation is crucial for determining whether the model is sufficient or requires further improvement. Additionally, evaluation using a confusion matrix provides a detailed overview of the model's performance by comparing predictions with actual data. In sentiment analysis, the confusion matrix includes the following components: True Positive (TP) — the number of correctly classified positive sentiment data, True Negative (TN) — correctly classified negative sentiment data, False Positive (FP) — negative sentiment data incorrectly classified as positive, and False Negative (FN) — positive sentiment data incorrectly classified as negative. This component is shown in Figure 2.
The evaluation formula based on the confusion matrix is as follows:

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (8)$$

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)$$

$$F1 - Score = 2\frac{\text{Precision .Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

Evaluation is conducted on the test data results to compare the performance of the SVM, Naïve Bayes, and Random Forest algorithms. This helps in understanding the strengths and weaknesses of each algorithm in effectively detecting positive and negative sentiment. The confusion matrix is a crucial tool for analyzing the distribution of model errors, particularly in unbalanced datasets.

Best Modelling

Best model selection is the process of identifying the most optimal model from a set of tested and evaluated models based on overall evaluation results. This step involves comparing different models using relevant metrics such as accuracy, precision, recall, and others to select the one that offers the best performance for the research objectives. Factors like computational speed, model interpretability, and the ability to handle imbalanced data are also taken into account. In some cases, ensemble methods or a combination of multiple models may be employed to enhance final performance. The selection of the best model ensures that the chosen solution is not only accurate but also efficient and reliable for practical use. The model with high accuracy and favorable metrics will be selected as the best model, ready for sentiment prediction in real-world applications.

III. RESULT AND DISCUSSION

Result

The dataset used in this research is divided into three main parts: training, validation, and testing data, ensuring that model evaluation is fair and representative. The data division is done using a stratification approach, which maintains the distribution of positive and negative sentiment labels across each subset. Training data is used to teach the model to recognize patterns in the data and optimize

model parameters. During the training process, the model adjusts its weights or parameters to minimize the error or loss function. At this stage, the model attempts to identify the relationship between input and output based on the available data. Validation data is used to monitor the model's performance during training and to fine-tune the model's hyperparameters. It offers an early evaluation of how well the model generalizes to unseen data and helps prevent overfitting by signaling whether the model is too specifically tailored to the training data. Testing data is used to evaluate the model's performance after training. It provides a final assessment of the model's ability to make accurate predictions or generalizations on completely new data that was not involved in training or validation. The results from testing data reveal how well the model is likely to perform in real-world applications.

The data split was stratified to maintain a balanced class distribution in each data subset, ensuring more accurate and reliable analysis results. This division is crucial for making sure that the developed model performs well not only on the data it was trained on but also on new, unseen data. The results of each data preprocessing step are presented in Table 2, Table 3, and Table 4. The data sharing results demonstrate that the sentiment distribution in each subset (training, validation, and testing) remains balanced, with similar proportions of positive and negative sentiments in all subsets.

Table 2. Training Data Prepocessing Result

Stages	Results	Description
User Reviews	3 Kali Telat meeting penting gara gara 3 kali transit pake lion air di bandara ini, service dan informasi sangat perlu diperbaiki	Raw text from user reviews complaining about airport delays and service quality.
Cleaning dan Case Folding	kali telat meeting penting gara gara kali transit pake lion air di bandara ini service dan informasi sangat perlu diperbaiki	Data cleaning by removing irrelevant characters and numbers and converting all letters to lowercase for text standards.
Normalization	kali terlambat meeting penting gara gara kali transit pakai lion air di bandara ini service dan informasi sangat perlu diperbaiki	Changed an unstandardized word from a typo such as "telat" to "terlambat". Using standardized word forms.
Stemming	kali lambat meeting penting gara gara kali transit pakai lion air di bandara ini service dan informasi sangat perlu baik	The process of reducing words to their base form, e.g. "terlambat" becomes "lambat", removing affixes to simplify the text.
Filtering dan Stopword	kali lambat meeting penting gara gara kali transit pakai lion air bandara service informasi sangat perlu baik	Removing words that do not provide information value such as "di", "dan", "yang" etc., thus clarifying the meaning.
Tokenization	['kali', 'lambat', 'meeting', 'penting', 'gara', 'gara', 'kali', 'transit', 'pakai', 'lion', 'air', 'bandara', 'service', 'informasi', 'sangat', 'perlu', 'baik']	The process of breaking down text into smaller units of words is easier to analyze in NLP.

Table 3. Validation Data Prepocessing Result

Stages	Results	Description
User Reviews	tidak diperbolehkan memakai troli kecil didalam gate, sehingga yang punya tentengan menjadi kewalahan dan capek..apalagi kalau brgkt dari gate ujung misal nya gate 1/2 .padahal dibandara lain mengijinkan untuk memakai troli kecil selama didalam gate	The raw text is a user complaint about the rule of not allowing small trolleys inside the gate, and the perceived experience of being overwhelmed and tired.
Cleaning dan Case Folding	tidak diperbolehkan memakai troli kecil didalam gate sehingga yang punya tentengan menjadi kewalahan dan capek apalagi kalau brgkt dari gate ujung misal gate padahal dibandara lain mengijinkan untuk memakai troli kecil selama didalam gate	Data cleaning is done by removing irrelevant characters and numbers and making all letters lowercase for text format consistency.
Normalization	tidak diperbolehkan memakai troli kecil di dalam gate sehingga yang punya tentengan menjadi kewalahan dan lelah apalagi jika berangkat dari gate ujung seperti gate padahal di bandara lain mengijinkan untuk memakai troli kecil selama di dalam gate	Correcting the spelling of words that are not standardized and harmonizing the writing of words that should be (e.g. "capek" becomes "lelah"). Text refinement to conform to Indonesian language standards.
Stemming	tidak boleh pakai troli kecil di dalam gate sehingga yang punya tenteng jadi kewalahan dan lelah apalagi jika berangkat dari gate ujung seperti gate padahal di bandara lain mengijinkan untuk pakai troli kecil lama di dalam gate	The process of reducing words to their base form, removing affixes. For example, "tentengan" becomes "tenteng" and "mengijinkan" becomes "ijinkan".
Filtering dan Stopword	tidak pakai troli kecil gate tenteng jadi kewalahan lelah berangkat gate ujung gate padahal bandara mengijinkan pakai troli kecil lama gate	Removing words that do not provide information value, such as "di", "dan", "yang", as well as words that are less important for further analysis.

Tokenization	['tidak', 'pakai', 'troli', 'kecil', 'gate', 'tenteng', 'jadi', 'kewalahan', 'lelah', 'berangkat', 'gate', 'ujung', 'gate', 'padahal', 'bandara', 'mengijinkan', 'pakai', 'troli', 'kecil', 'lama', 'gate']	The process of breaking text into the smallest units or tokens, which facilitates text analysis. Each word in a sentence is converted into a separate token.

Table 4. Test Data Prepocessing Result

Stages	Results	Description
User Reviews	Terlalu capek jalan kaki antar gate.	Raw text from users complaining of fatigue when walking between gates indicates a problem with the distance between gates at the airport.
Cleaning dan Case Folding	terlalu capek jalan kaki antar gate	Cleaning is done by removing unnecessary punctuation and ensuring all letters are lowercase for consistency.
Normalization	terlalu lelah jalan kaki antar gate	Change of nonstandard words to standardized words. “Capek” was changed to "lelah" to follow the more formal and standard Indonesian language.
Stemming	terlalu lelah jalan kaki antar gate	The stemming process changes the word to its base form. In this case, the word “lelah” remains unchanged because it is already in its base form, as well as other words.
Filtering dan Stopword	terlalu lelah jalan kaki antar gate	Removing words that have no further informative value. However, in this case no stopwords were removed as they were all considered important for analysis.
Tokenization	['terlalu', 'lelah', 'jalan', 'kaki', 'antar', 'gate']	The process of breaking a sentence into separate tokens. Each word or the smallest unit of the sentence is separated so that it can be analyzed further.

Table 5. Training Data Unigram and Bigram Results

Unigram	Bigram
['kali', 'lambat', 'meeting', 'penting', 'gara', 'gara', 'kali', 'transit', 'pakai', 'lion', 'air', 'bandara', 'service', 'informasi', 'sangat', 'perlu', 'baik']	['kali-lambat', 'lambat-meeting', 'meeting-penting', 'penting-gara', 'gara-gara', 'gara-kali', 'kali-transit', 'transit-pakai', 'pakai-lion', 'lion-air', 'air-bandara', 'bandara-service', 'service-informasi', 'informasi-sangat', 'sangat-perlu', 'perlu-baik']

Table 6. Validation Data Unigram and Bigram Results

Unigram	Bigram
['tidak', 'pakai', 'troli', 'kecil', 'gate', 'tenteng', 'jadi', 'kewalahan', 'lelah', 'brgkt', 'gate', 'ujung', 'gate', 'padahal', 'bandara', 'mengijinkan', 'pakai', 'troli', 'kecil', 'lama', 'gate']	['tidak-pakai', 'pakai-troli', 'troli-kecil', 'kecil-gate', 'gate-tenteng', 'tenteng-jadi', 'jadi-kewalahan', 'kewalahan-lelah', 'lelah-brgkt', 'brgkt-gate', 'gate-ujung', 'ujung-gate', 'gate-padahal', 'padahal-bandara', 'bandara-mengijinkan', 'mengijinkan-pakai', 'pakai-troli', 'troli-kecil', 'kecil-lama', 'lama-gate']

Table 7. Test Data Unigram and Bigram Results

Unigram	Bigram
['terlalu', 'lelah', 'jalan', 'kaki', 'antar', 'gate']	[‘terlalu-lelah’, ‘lelah-jalan’, ‘jalan-kaki’, ‘kaki-antar’, ‘antar-gate’]

(a) Positive (b) Negative

Figure 3. Word Cloud Unigram

(a) Positive (b) Negative

Figure 4. Word Cloud Bigram

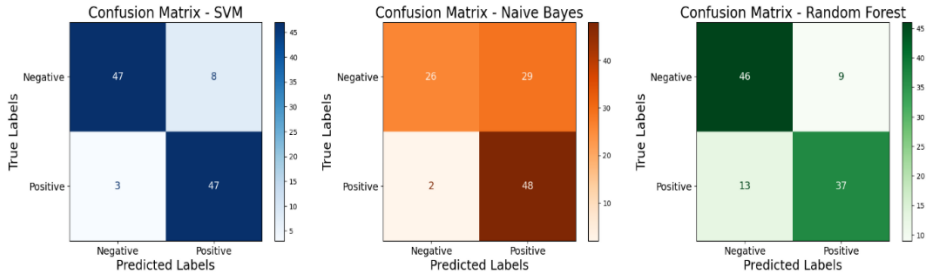
Next, a comparison of text feature representation using unigram and bigram is performed. This comparison aims to assess how the granularity of features (unigram vs. bigram) impacts the model's ability to understand and classify sentiment. Unigram refers to a feature representation that considers a single word as the unit of analysis, whereas bigram takes into account pairs of words that appear together in the text. This evaluation is crucial to determine whether more complex features, like bigrams, provide richer information and enhance model accuracy compared to simpler unigrams. The results of the Unigram and Bigram stages are presented in Table 5, Table 6, and Table 7.

Text features, represented using unigrams or bigrams, are then extracted using the TF-IDF method. TF-IDF is employed to measure the importance of a word in a document relative to the entire corpus. This method assigns more weight to words that appear frequently in a specific document but rarely in the entire corpus, helping the model capture more relevant information while reducing the impact of common or uninformative words. The Word Cloud for the Unigram representation can be seen in Figure 3, and the Word Cloud for the Bigram representation can be seen in Figure 4.

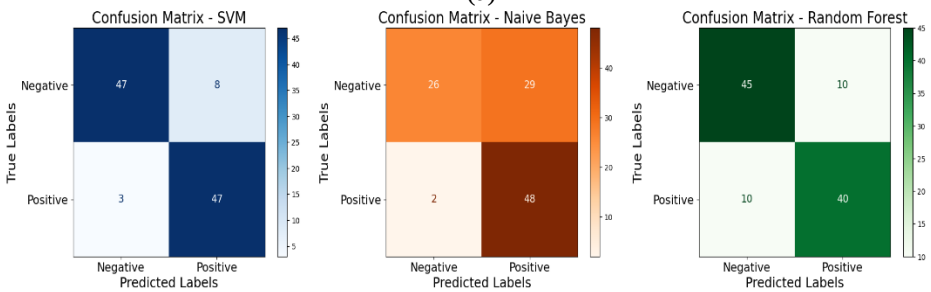
The Word Cloud visualization offers a clear depiction of the most frequently occurring words in the reviews for both unigram and bigram approaches. These frequently occurring words reveal dominant topics or themes within the dataset and provide insight into trends or patterns in user sentiment toward the reviewed product or service. In this analysis, the Word Cloud is used to identify keywords with high frequency and relevance in the context of each text feature representation.

At this stage, the model's performance was compared under two scenarios. In the first scenario, No Data Balancing, the model is trained using the data as is, without any adjustments or balancing between the positive and negative classes. This scenario aims to observe how the model performs when faced with an unbalanced data distribution. In the second scenario, Data Balancing, the minority class data is increased or the majority class data is decreased to achieve a more balanced distribution. This comparison seeks to evaluate whether data balancing can enhance the model's accuracy and its ability to classify data in a more equitable manner.

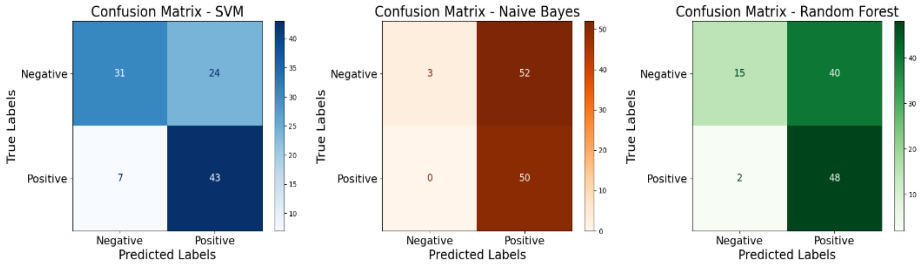
In the modeling stage, a comparison was made between three machine learning methods: SVM, Naïve Bayes, and Random Forest. This comparison aims to evaluate the performance of each method in handling pre-processed data and to identify the most effective algorithm for the classification task. The Confusion Matrix for each model is presented in Figure 5.



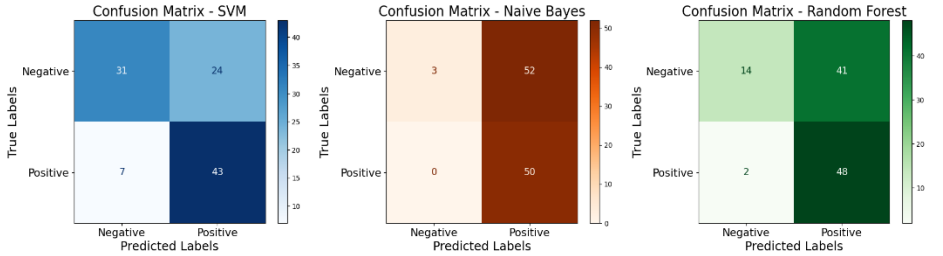
(a)



(b)



(c)



(d)

Figure 5. Confusion Matrix, (a) Unigram SMOTE, (b) Unigram No Balancing, (c) Bigram SMOTE, (d) Bigram No Balancing

Table 8. Results For Each Comparison Scenario

MODEL					Validation Data				Test Data			
Parameter				N-Gram	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Naïve Bayes SMOTE	Multinomial	Alpha	0.1	Unigram	81.9	82.6	81.9	81.9	80.9	81.0	80.9	80.8
			0.5		85.2	85.8	85.2	85.2	80.9	80.9	80.9	80.9
			1.0		**85.2**	**85.8**	**85.2**	**85.2**	**81.9**	**81.9**	**81.9**	**81.8**
			5.0		83.8	83.9	83.8	83.8	79.0	79.0	79.0	79.0
			10.0		84.2	84.3	84.2	84.3	80.9	80.9	80.9	80.9
			0.1	Bigram	72.3	72.4	72.3	72.4	71.4	71.4	71.4	71.3
			0.5		70.0	70.0	70.0	70.0	72.3	72.4	72.3	72.2
			1.0		70.0	70.0	70.0	70.0	72.3	72.4	72.3	72.2
			5.0		**70.9**	**71.0**	**70.9**	**70.9**	**74.2**	**74.3**	**74.2**	**74.1**
			10.0		70.9	71.0	70.9	70.9	74.2	74.3	74.2	74.1
Naïve Bayes Without Balancing Data	Multinomial	Alpha	0.1	Unigram	84.2	84.3	84.2	84.3	80.9	80.9	80.9	80.9
			0.5		**85.7**	**85.7**	**85.7**	**85.6**	**83.8**	**84.5**	**83.8**	**83.7**
			1.0		80.9	81.7	80.9	80.7	79.0	81.5	79.0	78.8
			5.0		70.4	77.2	70.4	67.8	70.4	78.3	70.4	68.8
			10.0		66.6	78.1	66.6	61.8	64.7	77.2	64.7	61.1
			0.1	Bigram	75.2	76.5	75.2	74.6	70.4	72.6	70.4	70.0
			0.5		**73.3**	**76.3**	**73.3**	**72.1**	**72.3**	**78.1**	**72.3**	**71.3**
			1.0		70.0	75.5	70.0	67.6	69.5	77.8	69.5	67.6
			5.0		53.8	63.5	53.8	39.4	50.4	75.7	50.4	36.7
			10.0		53.3	75.2	53.3	37.6	48.5	75.2	48.5	32.7
SVM SMOTE	Linear	C	0.01	Unigram	63.3	77.9	63.3	59.0	66.6	79.6	66.6	61.7
			0.1		82.3	84.5	82.3	82.2	79.0	80.4	79.0	78.6
			1		**89.0**	**89.1**	**89.0**	**89.0**	**92.3**	**92.6**	**92.3**	**92.3**
			10		86.1	86.2	86.1	86.1	86.6	86.9	86.6	86.6
			100		86.6	86.8	86.6	86.6	88.5	89.1	88.5	88.5
			0.01	Bigram	72.8	74.3	72.8	72.7	75.2	78.2	75.2	74.2
			0.1		**72.8**	**74.3**	**72.8**	**72.7**	**75.2**	**78.2**	**75.2**	**74.2**
			1		73.8	74.3	73.8	73.4	74.2	75.2	74.2	74.2
			10		73.8	75.3	73.8	73.0	64.7	67.0	64.7	64.1
			100		74.7	75.3	74.7	74.4	62.8	64.1	62.8	62.5

SVM Without Balancing Data	Linear	C	0.01	Unigram	52.8	27.9	552.8	36.5	47.6	22.6	47.6	30.7
			0.1		74.2	77.5	74.2	73.1	78.1	82.6	78.1	77.5
			1		**88.1**	**88.1**	**88.1**	**88.1**	**89.5**	**89.9**	**89.5**	**89.5**
			10		84.7	84.7	84.7	84.7	87.6	87.7	87.6	87.6
			100		86.1	86.2	86.1	86.1	85.7	86.4	85.7	85.7
			0.01	Bigram	52.8	27.9	52.8	36.5	47.6	22.6	47.6	30.7
			0.1		52.8	27.9	52.8	36.5	47.6	22.6	47.6	30.7
			1		**72.8**	**74.9**	**72.8**	**71.9**	**70.4**	**73.2**	**70.4**	**69.9**
			10		70.4	72.6	70.4	69.2	65.7	70.5	65.7	64.2
			100		70.4	72.6	70.4	69.2	65.7	70.5	65.7	64.2
Random Forest SMOTE	Classifier	n_estimators	10	Unigram	83.3	83.3	83.3	83.3	80.0	80.0	80.0	80.0
			50		85.2	85.2	85.2	85.2	78.1	78.1	78.1	78.0
			100		**85.2**	**85.2**	**85.2**	**85.2**	**80.0**	**80.1**	**80.0**	**79.9**
			200		85.2	85.2	85.2	85.2	77.1	77.1	77.1	77.1
			300		84.7	84.7	84.7	84.7	79.0	79.1	79.0	78.9
			10	Bigram	65.2	71.2	65.2	61.5	60.0	70.1	60.0	55.5
			50		65.2	71.2	65.2	61.5	60.9	70.8	60.9	56.9
			100		65.2	71.9	65.2	61.2	61.9	73.4	61.9	57.6
			200		65.7	71.6	65.7	62.1	61.9	73.4	61.9	57.6
			300		**66.6.**	**73.0**	**66.6**	**63.2**	**61.9**	**73.4**	**61.9**	**57.6**
Random Forest Tanpa Penyeimbang	Classifier	n_estimators	10	Unigram	84.2	84.3	84.2	84.3	77.1	77.2	77.1	77.1
			50		84.7	84.7	84.7	84.7	78.1	78.1	78.1	78.1
			100		83.3	83.3	83.3	83.3	79.0	79.0	79.0	79.0
			200		**84.2**	**84.2**	**84.2**	**84.2**	**81.9**	**81.9**	**81.9**	**81.9**
			300		84.2	84.2	84.2	84.2	79.0	79.1	79.0	79.0
			10	Bigram	67.6	72.5	67.6	64.9	60.0	68.4	60.0	56.1
			50		63.8	70.0	63.8	59.5	59.0	69.3	59.0	54.2
			100		63.8	70.7	63.8	59.2	60.9	72.8	60.9	56.3
			200		**64.2**	**72.0**	**64.2**	**59.6**	**63.8**	**71.6**	**63.8**	**58.9**
			300		63.8	71.6	63.8	58.9	59.0	74.2	59.0	52.8

Description: Bolded numbers indicate the best results for each process.

After all testing scenarios have been completed, the next step is to analyze the evaluation results to determine the best model. Comparing the performance of the tested models, including metrics such as accuracy, precision, recall, and F1-score can be seen in Table 8. The bolded and blue-colored numbers indicate the best result for each parameter, which represents the highest performance in the evaluation for that parameter in that model.

Analysis of the results showed significant differences between the use of Unigram and Bigram, as well as the effect of applying SMOTE in overcoming class imbalance compared to no data balancing. The use of Unigram proved to be superior in improving model performance compared to Bigram in almost all experiments, as seen from the better results on Accuracy, Precision, Recall, and F1-Score. For example, in Naïve Bayes with SMOTE, Unigram with Alpha = 0.5 yielded 85.2% Accuracy on the validation data, while Bigram only achieved 72.3%. Although Bigram can capture the context between words, the additional complexity it provides does not provide a significant improvement to the model's performance, especially on this dataset. On Naïve Bayes without balancing, Unigram achieved 85.7% accuracy, better than Bigram which only achieved 75.2%. Unigram is more effective without adding complexity.

A comparison between the use of SMOTE and without data balancing shows that the application of SMOTE improves model performance, especially in Recall, Precision, and F1-Score for minority classes. SMOTE addresses class imbalance by generating synthetic samples for minority classes that were previously under-detected. For example, in the Naïve Bayes model with SMOTE, Recall and F1-Score for the minority class increased despite a slight decrease in Accuracy. At Alpha = 0.5, Naïve Bayes with SMOTE resulted in an Accuracy of 85.2%, higher than the model without SMOTE which only achieved 84.2%. Similar results were seen in the SVM and Random Forest models, where SMOTE improved Recall and F1-Score, although Accuracy remained stable or slightly decreased. Overall, SMOTE helped the models become more balanced in handling both classes, resulting in more accurate and fair predictions, especially on data with class imbalance.

Each model showed the best performance at certain parameter combinations. On Naïve Bayes with SMOTE, the best results were achieved at Alpha = 0.5 with Unigram, resulting in 85.2% Accuracy on validation data and 81.0% on test data, with improved Recall and F1-Score for minority classes. On Naïve Bayes without balancing, Accuracy reached 85.7% on validation data and 80.9% on test data, but with no improvement on Recall. On SVM with SMOTE, the best results were achieved at C=1 with Unigram, resulting in 92.3% Accuracy on validation and test data, with superior performance on minority classes. On SVM without balancing, Accuracy was 88.1% on validation data and 89.5% on test data, but Recall was lower. On Random Forest with SMOTE, Accuracy was 85.2% on validation data and 80.0% on test data, stable on minority classes. In Random Forest without balancing, Accuracy is 84.2% in validation data and 81.9% in test data, but less optimal.

Overall, SVM with SMOTE was the best model of all tested. With the highest Accuracy of 92.3%, as well as significant improvements in Recall and F1-Score for the minority class, SVM with SMOTE successfully overcomes the class imbalance problem and provides more optimal results than the other models. This model successfully combines the strengths of SVM in text classification with the benefits of class balancing offered by SMOTE, making it the best choice for the dataset used.

Discussion

The SVM model with SMOTE proved to be the best model in sentiment analysis of Sultan Hasanuddin Airport reviews, with 92.3% Accuracy and significant improvement in Recall and F1-Score for minority classes. This indicates that SVM with SMOTE is not only effective in handling class imbalance, but also able to provide more balanced and fair results in sentiment classification, especially for reviews containing criticism or negative sentiments that are often overlooked. The application of SMOTE has a very positive impact on improving model performance on unbalanced datasets, such as reviews that tend to have more negative sentiment. With SMOTE, the model can produce more accurate and fairer predictions in handling reviews with minority sentiments, which were previously difficult for the model to detect without balancing. This is particularly relevant in the context of user reviews of public services, where feelings of dissatisfaction often outweigh positive reviews.

The impact of these results can be used directly for service improvement at Sultan Hasanuddin Airport. For example, if many reviews complain about the cleanliness of the facilities, airport managers can immediately improve cleaning services. Conversely, if there are problems in passenger flow efficiency, the data obtained from this model can provide more specific insights to make improvements. By using this model in real-time, airport managers can respond quickly to user complaints, improve customer experience, and drive higher passenger satisfaction. In addition, the results from this study show that SVM with SMOTE is not only effective in handling airport review data, but can also be applied to other platforms such as hotel review systems or product evaluations in e-commerce. This technique provides a more objective way of measuring public sentiment and can be used to analyze big data that is often imbalanced, helping companies or organizations make smarter data-driven decisions.

Comparison with previous studies using SMOTE or SVM on sentiment analysis in the public service sector shows comparable or even better results with the application of similar techniques. However, this research adds more value by applying these methods to highly imbalanced review data, as well as providing deeper insights into how these techniques can be used to improve service quality in the airline industry. Overall, this research not only contributes to sentiment analysis in the context of airports, but also opens up opportunities for the development of machine learning-based systems that better handle sentiment data in other public service sectors.

IV. CONCLUSION

This research successfully answers the problem formulation by showing that the Support Vector Machine (SVM) model with the application of SMOTE and Unigram configuration is the most effective model in the classification of public review sentiment towards Sultan Hasanuddin Airport in Makassar. With the highest accuracy of 92.3%, as well as high F1-score and Recall values for minority classes, this model is able to effectively handle the problem of data imbalance.

The use of Unigram consistently gives better results than Bigram, indicating that single word representation is more efficient and informative for this dataset. Meanwhile, the application of SMOTE was shown to improve the performance of the model, particularly in detecting less frequent negative

sentiments. This confirms that data balancing greatly affects the quality of predictions, especially on data with unbalanced class distributions.

Overall, the results of this study are in line with the stated objective of identifying the best method for machine learning-based sentiment analysis in the context of public service reviews. The findings are not only useful for the development of a sentiment monitoring system at Sultan Hasanuddin Airport, but also open up opportunities for similar applications in other public service sectors.

REFERENCES

[1] S. Ainin, A. Feizollah, N. B. Anuar, and N. A. Abdullah, "Sentiment analyses of multilingual tweets on halal tourism," *Tour. Manag. Perspect.*, vol. 34, no. February, p. 100658, 2020, doi: 10.1016/j.tmp.2020.100658.

[2] L. Zhang, M. Hou, Y. Liu, K. Wang, and H. Yang, "Measuring Beijing's international air connectivity and suggestions for improvement post COVID-19," *Transp. Policy*, vol. 116, no. November 2021, pp. 132–143, 2022, doi: 10.1016/j.tranpol.2021.11.015.

[3] S. Syafrizal, M. Afdal, and R. Novita, "Analisis Sentimen Ulasan Aplikasi PLN Mobile Menggunakan Algoritma Naïve Bayes Classifier dan K-Nearest Neighbor," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. 1, pp. 10–19, 2023, doi: 10.57152/malcom.v4i1.983.

[4] D. Ramadhansyah, A. Asrofiq, and Y. Yunefri, "Analisis Sentimen Ulasan penumpang maskapai penerbangan di Indonesia… ZONAsi," *J. Sist. Inf.*, vol. 6, no. 2, pp. 287–297, 2024.

[5] K. Airports, "Kja em," pp. 103–110, 2023.

[6] K. D. M. Muhammad Hidayat Nasmi, "Jurnal Ilmu Administrasi Publik," *J. Ilmu Adm. Publik*, vol. 6, no. 2, pp. 165–175, 2021.

[7] S. Khomsah, "Naive Bayes Classifier Optimization on Sentiment Analysis of Hotel Reviews," *J. Penelit. Pos dan Inform.*, vol. 10, no. 2, p. 157, 2020, doi: 10.17933/jppi.2020.100206.

[8] S. S. Aljameel *et al.*, "A sentiment analysis approach to predict an individual's awareness of the precautionary procedures to prevent covid-19 outbreaks in Saudi Arabia," *Int. J. Environ. Res. Public Health*, vol. 18, no. 1, pp. 1–12, 2021, doi: 10.3390/ijerph18010218.

[9] Y. Dzyuban *et al.*, "Sentiment Analysis of Weather-Related Tweets from Cities within Hot Climates," *Weather. Clim. Soc.*, vol. 14, no. 4, pp. 1133–1145, 2022, doi: 10.1175/WCAS-D-21-0159.1.

[10] Rabiatul Adawiyah and Munifah, "Eksplorasi Kapasitas Pengkodean Amplitudo Untuk Model Quantum Machine Learning," *Inform. J. Tek. Inform. dan Multimed.*, vol. 3, no. 1, pp. 38–58, 2023, doi: 10.51903/informatika.v3i1.232.

[11] A. A. Dhani, T. A. Y. Siswa, and W. J. Pranoto, "Perbaikan Akurasi Random Forest Dengan ANOVA Dan SMOTE Pada Klasifikasi Data Stunting," *Teknika*, vol. 13, no. 2, pp. 264–272, 2024, doi: 10.34148/teknika.v13i2.875.

[12] E. Sur and H. Çakır, "Bibliometric analysis of the use of sentiment analysis in the context of service quality," *Yalvaç Akad. Derg.*, vol. 8, no. 1, pp. 81–104, 2023, doi: 10.57120/yalvac.1258627.

[13] A. H. Salman and W. A. M. Al-Jawher, "Performance Comparison of Support Vector Machines, AdaBoost, and Random Forest for Sentiment Text Analysis and Classification," *J. Port Sci. Res.*, vol. 7, no. 3, pp. 300–311, 2024, doi: 10.36371/port.2024.3.8.

[14] Viny Gilang Ramadhan and Yuliant Sibaroni, "Sentiment Analysis of Public Opinion Related to Rapid Test Using LDA Method," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 4, pp. 672–679, 2021, doi: 10.29207/resti.v5i4.3139.

[15] M. O. Odim, A. O. Ogunde, B. O. Oguntunde, and S. A. Phillips, "Exploring the performance characteristics of the naïve bayes classifier in the sentiment analysis of an Airline's social media data," *Adv. Sci. Technol. Eng. Syst.*, vol. 5, no. 4, pp. 266–272, 2020, doi: 10.25046/aj050433.

[16] A. A. Syahfitri, M. F. Mulya, P. D. Larasati, and S. Anwar, "Perancangan Sistem Informasi Penerimaan Peserta Didik Baru Menggunakan Codeigniter, Bootstrap Dan Mysql (Studi Kasus: Raudhatul Athfal Az-Zahra)," *J. SISKOM-KB (Sistem Komput. dan Kecerdasan Buatan)*, vol. 6, no. 2, pp. 125–136, 2023, doi: 10.47970/siskom-kb.v6i2.375.

[17] M. B. Ressan and R. F. Hassan, "Naïve-Bayes family for sentiment analysis during COVID-19 pandemic and classification tweets," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 28, no. 1, pp. 375–

383, 2022, doi: 10.11591/ijeecs.v28.i1.pp375-383.
[18] N. Jalal, A. Mehmood, G. S. Choi, and I. Ashraf, "A novel improved random forest for text classification using feature ranking and optimal number of trees," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2733–2742, 2022, doi: 10.1016/j.jksuci.2022.03.012.
[19] A. Sagita, A. Faqih, G. Dwilestari, B. Siswoyo, and D. Pratama, "Penerapan Metode Random Forest Dalam Menganalisis Sentimen Pengguna Aplikasi Capcut Di Google Play Store," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 6, pp. 3307–3313, 2024, doi: 10.36040/jati.v7i6.8205.
[20] A. M. Sarhan, H. Ayman, M. Wagdi, B. Ali, A. Adel, and R. Osama, "Integrating machine learning and sentiment analysis in movie recommendation systems," *J. Electr. Syst. Inf. Technol.*, 2024, doi: 10.1186/s43067-024-00177-7.
[21] E. H. Muktafin and P. Kusrini, "Sentiments analysis of customer satisfaction in public services using K-nearest neighbors algorithm and natural language processing approach," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 19, no. 1, pp. 146–154, 2021, doi: 10.12928/TELKOMNIKA.V19I1.17417.
[22] Hermanto, A. Y. Kuntoro, T. Asra, E. B. Pratama, L. Effendi, and R. Ocanitra, "Gojek and Grab User Sentiment Analysis on Google Play Using Naive Bayes Algorithm and Support Vector Machine Based Smote Technique," *J. Phys. Conf. Ser.*, vol. 1641, no. 1, 2020, doi: 10.1088/1742-6596/1641/1/012102.
[23] R. S. Y. Zebua, *Fenomena artificial intelleigence*, no. June. 2023.