khazanah informatika

Jurnal Ilmu Komputer dan Informatika

Using SVM and KNN for Predicting Customer Response Sentiment of M-PAJAK Application

Muhammad Titan Rama Adi Wijaya [1], Ida Widaningrum [2*], Angga Prasetyo [3], Dyah Mustikasari [4]

[1,2,3,4]Department of Informatics Engineering
Universitas Muhammadiyah Ponorogo
Ponorogo
* correspondence: iwidaningrum.as@gmail.com

Abstract- M-Pajak, an application initiated by the Directorate General of Taxes, signifies the modernization of taxation and serves a crucial function. This application facilitates taxpayers in meeting their tax obligations. User satisfaction with this application may be assessed via reviews on the Google Play Store. While this application fulfills client satisfaction, its sustained success is significantly contingent upon user contentment and experience. Sentiment analysis is essential for elucidating user evaluations and interactions with the program. This research analyses the sentiment of M-Pajak application reviews on Google Play using Support Vector Machine (SVM) and K-Nearest Neighbour (KNN), supported by the Term Frequency-inverse Document Frequency (TF-IDF) feature extraction method. A total of 1000 reviews between December 11, 2022 and December 2, 2023 were processed using KNN and SVM. The KNN algorithm yielded 153 positive predictions and 847 negative predictions and achieved 94% of accuracy. Meanwhile, SVM achieved an accuracy of 88.10%, with 325 positive predictions and 675 negative predictions. The results demonstrate the superiority of KNN in sentiment classification of M-Pajak reviews. This study also indicates that negative comments outnumber positive ones in this application. This serves as a signal for the Directorate General of Taxation to enhance user satisfaction with the M-Pajak application through continuous updates.

Keywords: Sentiment Analysis; K-Nearest Neighbour; Support Vector Machine; M-Pajak

Article info: *submitted March 12, 2024, revised April 23, 2025, accepted July 19, 2025*

1. Introduction

An opinion is a response, a judgment, or the result of an individual's subjective beliefs regarding anything that is not universally acknowledged as true. Concerning a topic or occurrence, individuals possess varying perspectives [1]. In the modern era, opinions and viewpoints influence almost every human behavior. People look to others for their opinions when they are trying to decide or make a determination. For instance, while deciding whether or not to use or download an application from the app store, a person will consider other users' thoughts and reviews before making that decision. Developers and allied institutions rely heavily on user feedback to understand how well applications function.

Sentiment analysis, sometimes referred to as feeling analysis, is an automated procedure that extracts, interprets, and processes text input in order to gather information [2]. Opinions can be gleaned from a variety of sources, including social media, blogs, comments, documents, and questionnaires, by using sentiment analysis or opinion mining. Sentiment analysis enables the monitoring of customer satisfaction over a product.

The Directorate General of Taxes (DJP) created M-Pajak, a mobile application that leverages the jknmobile.com platform, with the intention of offering taxpayers more individualized, straightforward, and effective services [3]. In addition to other data collected by the application distribution process, M-Pajak produces data pertaining to taxes. M-Pajak is accessible via digital platforms, specifically the Google Play Store and the Apple App Store. Over a million people have downloaded the M-Pajak application from Google Play, and thanks to the capabilities offered by Google Play, each user is able to leave a review for the program [4]. Users have unrestricted access to these reviews. If appropriately processed, the 4,000 M-Pajak application evaluation data points may be valuable. Since user feedback is the best source of criticism and ideas, data processing findings will be used to improve and develop applications. The opinions expressed by users on Google Play may have an impact on prospective users' decisions to utilize particular programs. Potential consumers' decisions to utilize particular applications can be influenced by every comment posted by every user on Google Play. With the volume of review data already in existence, processing it by hand will be challenging. As a result, it is imperative to automatically monitor user feedback—whether favorable or negative—about the program. Given the significance of user evaluations for an application's sustainability, it would be

preferable to perform research using M-Pajak application review data from Google Play as a sample of research data processing.

The goal of grouping new data based on the majority of categories from the K training data that are closest to it—also referred to as "nearest neighbor"—is achieved by processing data using two classification methods, namely the K-Nearest Neighbor (KNN) Training Algorithm. Classifying fresh data according to characteristics and training samples is the primary goal of this method [5]–[7]. Support Vector Machine (SVM) is the next machine learning method used for regression and classification. Structural risk reduction is the foundation of SVM, which separates the input space into two classes using a hyperplane to process data [8]. The Term Frequency-inverse Document Frequency (TF-IDF) approach can be used to complete the task. The methodology utilized, may become more accurate if the above strategy is selected, particularly when training KNN and SVM [9]. The use of SVM to measure sentiment is also explored in studies [10]–[12], and the KNN algorithm is applied in [13]–[15].

This study uses the KNN and SVM algorithms to view M-Pajak application reviews and sort them into good and negative groups. We use Python and Flask to carry out the procedure, and we use TF-IDF to give different weights to words. The goal is to find out how strong people's sentiment is towards M-Pajak and then use the data to investigate the application and tax-related issues in more depth.

2. Methods

The research comprises nine stages: data retrieval, data preprocessing, labeling, TF-IDF weighting, training and testing data creation, K-Nearest Neighbor and Support Vector Machine classification, assessment, and data presentation. The research stages are depicted in Figure 1.

The system design comprises a login page, dashboard, data import, preprocessing, weighting, labeling, and KNN and SVM classification methods. Data was collected by reviewing M-Pajak application data through the Google Play website, utilizing web scraping techniques from web pages coded with a markup language. Google Play is a versatile program that is compatible with both mobile devices and websites [16]. This process utilizes the Python programming language in Google Colab to generate a CSV file. The data is primarily sourced from the M-Pajak application's data, ratings, and reviews. The data was collected between December 11, 2022 and December 2, 2023. The dataset comprises 1000 review comments. Comment data is a written evaluation that will be utilized for sentiment analysis at a later time. During preprocessing for sentiment analysis, optimization is achieved through four stages: case folding, tokenizing, stopword removal, and stemming [17]. This labeling is done using a lexicon-based technique in Python. If the score is less than 0, the sentiment is negative; if the score is more than 0, the sentiment is positive. The TF-IDF weighting procedure starts by preparing the necessary data through preprocessing data that was saved earlier. Subsequently, utilize pandas to read the file, specifying solely the label and review fields. We calculate TF-IDF using the Scikit-Learn Python library and the TfidfVectorizer module, resulting in the term with the highest term frequency being identified. Dividing preprocessed data into a ratio of 8:2 results in 20% test data (200 data) and 80% training data (800 data)) [18].

$$tfidf(word) = tf(word) * log \frac{N}{df(word)} \quad (1)$$

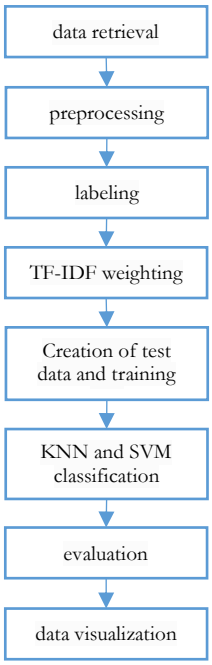


Figure 1. Research Methodology

TF-IDF is a text processing technique that assigns weights to terms in documents [19], [20]. In this case, tf is the number of times word t appears in the document, df is the number of documents that contain word t, and N is the total number of documents in the dataset [21]. This approach seeks to look for or obtain intriguing words [9]. Term Frequency (TF) is the count of words in a corpus or collection of writing. IDF, or Inverse Document Frequency, refers to the frequency of phrases appearing in a corpus [22].

The KNN technique is utilized in classification to determine the similarity or distance between testing and training samples by identifying the nearest neighbors of K testing and training samples, which are then changed based on the categories of these neighbors [23]–[28]. The Scikit-learn library is utilized to implement the KNN and SVM algorithms in Python, employing an 80:20 training-to-testing data ratio. The KNN classification function operates by supplying training data to produce predictions on the testing data, which are then returned as an array. The SVM algorithm optimizes hyperplane borders by sorting the data and finding the nearest data margin distance from each class [29],[30].

The evaluation phase assesses data classification utilizing the KNN and SVM methods through the confusion matrix method to determine the accuracy of the algorithm [31]–[33]. Lastly, Data Visualization. Once the review data has been categorized, Matplotlib in Python will be used to visualize the amount of predictions from good and bad reviews [34]–[38].

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (2)$$

In this formula (2), TP (True Positive) represents the number of positive data predicted as accurate, while TN (True Negative) represents the number of negative data predicted as correct. FP (False Positive): The total number of negative data that is predicted as positive (incorrect), and FN (False Negative): The total number of positive data that is predicted as negative (incorrect) [39].

3. Result and Discussion

The sentiment analysis procedure for utilizing the M-Pajak program involves data collecting, preprocessing, labeling, weighting, categorizing the data into training and testing datasets, applying KNN and SVM algorithms, and concluding with evaluation.

1. *Data retrieval*

Data collection utilizes web scraping techniques on websites that utilize a markup language. The procedure is conducted using the Python programming language in Google Colab, resulting in the generation of a CSV file. Data was gathered from 1,000 sample reviews out of a total of 4,000 reviews on the M-Pajak application, collected between December 11, 2022, and December 2, 2023.

The imported data originates from a CSV file with an ID column represented by numbers, a score column represented by labels, and a content column represented by text or reviews for data processing (Refer to Figure 2).

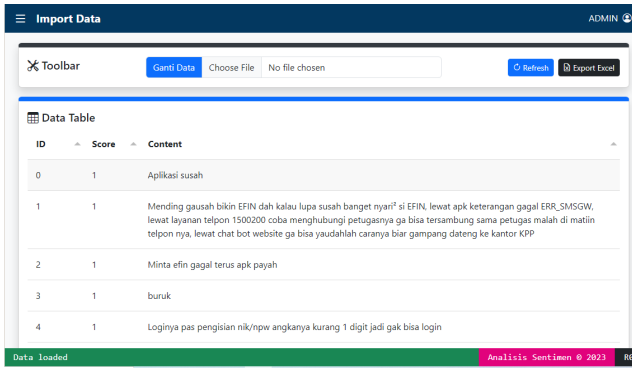


Figure 2. Import data from CSV file

2. *Preprocessing*

The raw data undergoes preprocessing to transform it into clean data suitable for utilization. The preprocessing method includes case folding, removing stopwords, tokenizing, and stemming as shown in Table 1.

Table 1. Preprocessing

Preprocessing Stages	Text Transformation Results
Text	Susahnya mau bayar pajak, terlalu berbelit
Case Folding	susahnya mau bayar pajak terlalu berbelit
Stopwords Removal	susahnya bayar pajak berbelit
Tokenize	'susahnya', 'bayar', 'pajak', 'berbelit'
Stemming	susah bayar pajak belit

During case folding, capital characters "A" to "Z" in the data are converted to lowercase letters "a" to "z". Stopwords are often occurring terms that are deemed insignificant in big quantities, such as "dan", "yang", "dari", "di", etc. The objective of using it is to enhance the text's focus on key words by removing less essential ones. Tokenizing involves dividing text into segments known as "tokens" for analysis, such as "numbers," "words," "punctuation," "symbols," and other significant elements. Tokenization can be used to paragraphs or sentences, however in NLP, tokens are specifically considered as "words". Stemming is the last stage of data preprocessing when words with affixes are transformed into their base forms by removing prefixes, infixes, and suffixes. Stemming is utilized on Indonesian language evaluations using literary libraries within the Python programming environment. Non-existent words will be eliminated from the literature.

3. *TF-IDF (Term Frequency-Inverse Document Frequency)*

The TF-IDF weighting process begins by preparing the required data using previously saved data preprocessing. Then, read the file using pandas, which only requires the label and review fields. To calculate TF-IDF from our data, we use the Scikit-Learn Python library with the TfidfVectorizer module, the result of which will be the top terms with the largest term frequency (Figure 3).

```
#calculate tf
for i in range(len(sorted_terms)):
    sorted_terms[i][1]['tf'] = sorted_terms[i][1]['count'] / len(sorted_terms)

#calculate idf
for i in range(len(sorted_terms)):
    sorted_terms[i][1]['idf'] = np.log(len(df) / sorted_terms[i][1]['count'])

#calculate tf-idf
for i in range(len(sorted_terms)):
    sorted_terms[i][1]['tf-idf'] = sorted_terms[i][1]['tf'] * sorted_terms[i][1]['idf']

return sorted_terms
```

Kata	Frequency	Tf	IDF	TF-IDF
aplikasi	240	0.3342618384401114	1.4017985476558559	0.46856775966212455
pajak	188	0.2618384401114206	1.645995508167898	0.4309848962890875

Figure 3. Word Weighting Process

4. *Classification*

The classification stage is the primary phase in the sentiment analysis process. The classification process utilizes data that has been split into training data and test data. The training procedure

will generate a classifier model that will be stored in a file in the shape of a pickle. A pickle is a feature vector created by combining each feature word with its corresponding sentiment label. The next step involves the testing of the data using the previously developed classifier model. The end outcome of this procedure is the sentiment expressed in the review document, which can be either good or negative.

The K-Nearest Neighbor (KNN) technique determines the category of a test sample by calculating the distance or similarity between the test sample and all training samples, identifying the K nearest neighbors of the test sample in the training set, and assigning the category based on these neighbors' categories. Support Vector Machine (SVM) maximizes the margins of the hyperplane that divides a dataset. Hyperplane computations involve determining the margin distance to the nearest data point from each class. Implementing KNN and SVM in Python involves invoking the classification functions for KNN and SVM, providing training data, and generating predictions from test data, resulting in an array of outcomes (Figure 4-5).

```
def knn():
    knn_df = df.copy()

    sorted_terms = tfidf(combined_content)

    pbar = tqdm(total=len(knn_df))

    for i in range(len(knn_df)):
        pbar.update(1)
        sentiment = 0

        for word in knn_df['cleaned'][i].split():
            for j in range(len(sorted_terms)):
                #if word is in both lexicon and sorted_terms
                if word == sorted_terms[j][0] and word in lexicon['word'].values:
                    sentiment += lexicon['weight'][lexicon['word'] == word].values[0] * sorted_terms[j][1]['tf-idf']

        knn_df['sentiment'][i] = sentiment

    #label data, if score is 3 and below, then it is negative, else it is positive
    knn_df['label'] = knn_df['score'].apply(lambda x: 0 if x <= 3 else 1)

    #and modify score by multiplying it with sentiment
    knn_df['score'] = knn_df['score'].mul(knn_df['sentiment'])

    X = knn_df[['score', 'sentiment']]
    y = knn_df['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

    knn = KNeighborsClassifier(n_neighbors=2)
    knn.fit(X_train, y_train)

    y_pred = knn.predict(X_test)

    print('Accuracy: ', accuracy_score(y_test, y_pred))
```

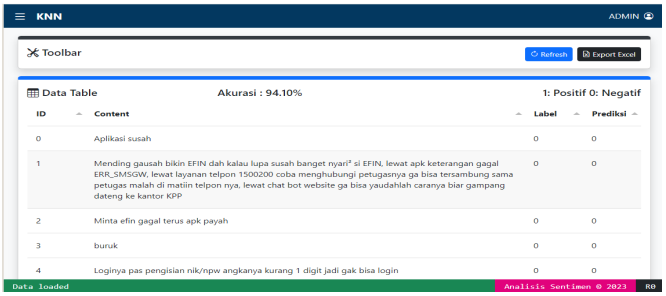
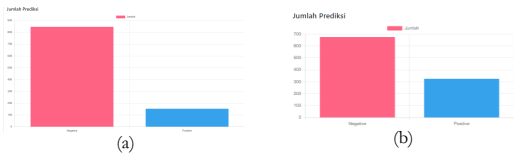


Figure 4. K-Nearest Neighbor Classification Process

```
def svme():
    svm_df = df.copy()

    sorted_terms = tfidf(combined_content)

    pbar = tqdm(total=len(svm_df))

    for i in range(len(svm_df)):
        pbar.update(1)
        sentiment = 0

        for word in svm_df['cleaned'][i].split():
            for j in range(len(sorted_terms)):
                #if word is in both lexicon and sorted_terms
                if word == sorted_terms[j][0] and word in lexicon['word'].values:
                    sentiment += lexicon['weight'][lexicon['word'] == word].values[0] * sorted_terms[j][1]['tf-idf']

        svm_df['sentiment'][i] = sentiment

    #label data, if score is 3 and below, then it is negative, else it is positive
    svm_df['label'] = svm_df['score'].apply(lambda x: 0 if x <= 3 else 1)

    #and modify score by multiplying it with sentiment
    svm_df['score'] = svm_df['score'].mul(svm_df['sentiment'])

    X = svm_df[['score', 'sentiment']]
    y = svm_df['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

    svme = svm.SVC(kernel='linear')
    svme.fit(X_train, y_train)

    y_pred = svme.predict(X_test)

    print('Accuracy: ', accuracy_score(y_test, y_pred))
```

Figure 5. Support Vector Machine Classification Process

Prediction results using KNN and SVM are shown in Figure 6.



(a) (b)

Figure 6. Number of Predictions (a) KNN and (b) SVM

Figure 6 shows that the number of predictions is 152 positive data and 848 negative data for KNN, and the number of predictions is 325 positive data, and 675 negative data for SVM.

5. *Evaluation*

Test evaluation of data classification using the Support Vector Machine and K-Nearest Neighbor methods using the confusion matrix method as shown in Table 2, with accuracy results of 94.10% for KNN and 88.10% for SVM.

Tabel 2. Accuracy with confussion matrix

N = 1000 (data training: data testing=80:20)	Predicted No		Predicted Yes		Accuracy	
	KNN	SVM	KNN	SVM	KNN	SVM
Actual No	58	2	1	117	94.10%	88.10%
Actual Yes	790	673	151	208		

Based on KNN and SVM accuracy testing, data on the number of M-Pajak data predictions was obtained as in Figure 6.

Figure 7. Comparison results of KNN and SVM algorithms

From Figure 7, it can be seen that KNN produces an accuracy of 94.10% and a total of 152 positive data and 848 negative data predictions. Meanwhile, SVM produces an accuracy of 88.10% and a total of 325 positive data and 675 negative data predictions. The difference between the KNN test results being better than SVM is because the value of the number of correct predictions is 941 data while SVM has 881 correct prediction data. The difference in test scores shows that KNN is better than SVM on this dataset. One reason KNN is better is because it can work better with smaller datasets. This is because KNN only needs to know the distance between data points, so it does not require a complicated training process [40]. On the other hand, SVM often requires a lot of data and a balanced class distribution to create a perfect separating hyperplane [41]. In this case, KNN performs better because the dataset has little data and there is likely to be an imbalance in the classes.

Testing training data and testing data with a data ratio of 80: 20 out of 1000 data obtained after 10x testing obtained accuracy as in Table 3.

Table 3. Training and Testing Data Testing

Testing	KNN Accuracy	SVM Accuracy
1	94	88,10
2	94,10	87,90
3	93,80	87,90
4	93,70	88,10
5	93,90	88,10
6	94,10	87,90
7	94,10	88,10
8	93,90	88,10
9	93,70	88,10
10	93,50	87,90

From Table 3, it is known that the highest accuracy of KNN and SVM is 94.10% and 88.10%, while the lowest accuracy testing values are 93.50% and 87.90%.

4. Conclusion

Sentiment analysis was conducted on public opinion regarding the performance of the m-tax application system using K-Nearest Neighbor (KNN) and Support Vector Machine (SVM) algorithms with a dataset of 1000 entries. Algorithms in sentiment analysis, namely KNN, achieve an accuracy of 94.10% with 153 positive data and 847 negative data predicted. SVM achieved an accuracy of 88.10% with 325 positive data predictions and 675 negative data predictions. There are more negative predictions than positive ones for both the KNN and SVM algorithms, suggesting that the M-Pajak application needs further enhancement. The sentiment analysis application resulted in superior KNN test outcomes compared to SVM. The number of valid predictions for KNN is 941, while SVM has 881 correct predictions.

Despite the completion of this research, several limitations remain that warrant improvement. Increasing the dataset size is recommended to enhance accuracy, along with conducting further experiments using varied training and test sets to improve model performance. The use of preprocessing techniques such as normalization and comprehensive evaluation metrics like precision, recall, and F1-score is also advised. Future studies may explore alternative classification methods and apply statistical analysis to strengthen both the practical and scientific contributions of the research.

Based on the results of the analysis showing the dominance of negative comments, further research is needed to identify aspects that need to be developed, both in applications, tax services, and others. A more in-depth analysis of the content of user comments is needed to find out the specific source of dissatisfaction.

Reference

[1] D. N. I. Huda and C. Prianto, "Analisis Sentimen Layanan Jasa Pengiriman Pada Ulasan Play Store : Systematic Literature Review," *J. iInformatika dan Teknol. Komput.*, vol. 04, no. 02, pp. 87–98, 2023.

[2] S. S. Salim and J. Mayary, "Analisis Sentimen Pengguna Twitter Terhadap Dompet Elektronik Dengan Metode Lexicon Based Dan K – Nearest Neighbor," *J. Ilm. Inform. Komput.*, vol. 25, no. 1, pp. 1–17, 2020, doi: 10.35760/ik.2020.v25i1.2411.

[3] Nora Galuh Candra Asmarani, "Apa itu M-Pajak?," *https://news.ddtc.co.id/*, Nov. 2021. .

[4] Google, "M-Pajak Google Play," *https://play.google.com/store/apps/details?id=id.go.pajak.djp*. .

[5] L. Legito *et al.*, "Penerapan Algoritma K-Nearest Neighbor untuk Analisis Sentimen Terhadap Isu Khilafah dan Radikalisme di Indonesia: Implementation K-Nearest Neighbor Algorithm for Sentiment Analysis on Khilafah and Radicalism Issues in Indonesia," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 3, no. 2, pp. 324–330, 2023.

[6] A. R. Isnain, J. Supriyanto, and M. P. Kharisma, "Implementation of K-Nearest Neighbor (K-NN) Algorithm For Public Sentiment

Analysis of Online Learning," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 15, no. 2, pp. 121–130, 2021.

[7] I. Triguero, J. Maillo, J. Luengo, S. García, and F. Herrera, "From big data to smart data with the k-nearest neighbours algorithm," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016, pp. 859–864.

[8] Binus University, "Support Vector Machine Algorithm," *https://sis.binus.ac.id/2022/02/14/support-vector-machine-algorithm/*, 2022. .

[9] R. Wati, S. Ernawati, and H. Rachmi, "Pembobotan TF-IDF Menggunakan Naïve Bayes pada Sentimen Masyarakat Mengenai Isu Kenaikan BIPIH," *J. Manaj. Inform.*, vol. 13, no. 1, pp. 84–93, 2023, doi: 10.34010/jamika.v13i1.9424.

[10] H. Sulastomo, R. Ramadiansyah, K. Gibran, E. Maryansyah, and A. Tegar, "Analisis Sentimen Pada Twitter@ Ovo_Id dengan Metode Support Vectore Machine (SVM)," *J-SAKTI (Jurnal Sains Komput. dan Inform.*, vol. 6, no. 2, pp. 1050–1056, 2022.

[11] B. A. Maulana, M. J. Fahmi, A. M. Imran, and N. Hidayati, "Analisis Sentimen Terhadap Aplikasi Pluang Menggunakan Algoritma Naive Bayes dan Support Vector Machine (SVM): Sentiment Analysis of Pluang Applications With Naive Bayes and Support Vector Machine (SVM) Algorithm," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. 2, pp. 375–384, 2024.

[12] D. Atmajaya, A. Febrianti, and H. Darwis, "Metode SVM dan Naive Bayes untuk Analisis Sentimen ChatGPT di Twitter," *Indones. J. Comput. Sci.*, vol. 12, no. 4, 2023.

[13] K. Mustaqim, F. A. Amaresti, and I. N. Dewi, "Analisis Sentimen Ulasan Aplikasi PosPay untuk Meningkatkan Kepuasan Pengguna dengan Metode K-Nearest Neighbor (KNN)," *Edumatic J. Pendidik. Inform.*, vol. 8, no. 1, pp. 11–20, 2024.

[14] S. D. Prasetyo, S. S. Hilabi, and F. Nurapriani, "Analisis Sentimen Relokasi Ibukota Nusantara Menggunakan Algoritma Naïve Bayes dan KNN," *J. KomtekInfo*, pp. 1–7, 2023.

[15] P. Astuti and N. Nuris, "Penerapan Algoritma KNN Pada Analisis Sentimen Review Aplikasi Peduli Lindungi," *Comput. Sci.*, vol. 2, no. 2, pp. 137–142, 2022.

[16] S. A. Aaputra, D. Rosiyadi, W. Gata, and S. M. Husain, "Sentiment Analysis Analysis of E-Wallet Sentiments on Google Play Using the Naive Bayes Algorithm Based on Particle Swarm Optimization," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 3, pp. 377–382, 2019.

[17] K. I. Ruslim, P. P. Adikara, and Indriati, "Analisis Sentimen Pada Ulasan Aplikasi Mobile Banking Menggunakan Metode Support Vector Machine dan Lexicon Based Features," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 7, pp. 6694–6702, 2019.

[18] A. I. Tanggraeni and M. N. N. Sitokdana, "Analisis Sentimen Aplikasi E-Government pada Google Play Menggunakan Algoritma Naïve Bayes," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 9, no. 2, pp. 785–795, 2022, doi: 10.35957/jatisi.v9i2.1835.

[19] A. Mee, E. Homapour, F. Chiclana, and O. Engel, "Sentiment analysis using TF–IDF weighting of UK MPs' tweets on Brexit," *Knowledge-Based Syst.*, vol. 228, p. 107238, 2021.

[20] G. Domeniconi, G. Moro, R. Pasolini, and C. Sartori, "A comparison of term weighting schemes for text classification and sentiment analysis with a supervised variant of tf. idf," in *Data Management Technologies and Applications: 4th International Conference, DATA 2015, Colmar, France, July 20-22, 2015, Revised Selected Papers 4*, 2016, pp. 39–58.

[21] C. A. N. Agustina, R. Novita, and N. E. Rozanda, "The implementation of TF-IDF and Word2Vec on booster vaccine sentiment analysis using support vector machine algorithm," *Procedia Comput. Sci.*, vol. 234, pp. 156–163, 2024.

[22] N. Arifin, U. Enri, and N. Sulistiyowati, "Penerapan Algoritma Support Vector Machine (SVM) dengan TF-IDF N-Gram untuk Text Classification," *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 6, no. 2, p. 129, 2021, doi: 10.30998/string.v6i2.10133.

[23] M. R. Huq, A. Ahmad, and A. Rahman, "Sentiment analysis on Twitter data using KNN and SVM," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, 2017.

[24] H. Wisnu, M. Afif, and Y. Ruldevyani, "Sentiment analysis on customer satisfaction of digital payment in Indonesia: A comparative study using KNN and Naïve Bayes," in *Journal of Physics: Conference Series*, 2020, vol. 1444, no. 1, p. 12034.

[25] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "An k NN model-based approach and its application in text categorization," in *Computational Linguistics and Intelligent Text Processing: 5th International Conference, CICLing 2004 Seoul, Korea, February 15-21, 2004 Proceedings 5*, 2004, pp. 559–570.

[26] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, 2003, pp. 986–996.

[27] M. Jannah, A. A. Nababan, and Y. S. Ningsi, "Penerapan Metode K-Nearest Neighbor Dalam Identifikasi Jenis Ikan Salmon Yang Dapat Dikomsumsi Untuk Bahan Mpasi Bayi," *J. Teknol. Sist. Inf. dan Sist. Komput. TGD*, vol. 6, no. 2, pp. 636–644, 2023.

[28] M. Hafizh Mahendra, D. Triantoro Murdiansyah, and K. Muslim Lhaksmana, "Analisis Sentimen Tweet COVID-19 Menggunakan Metode K-Nearest Neighbors dengan Ekstraksi Fitur TF-IDF dan CountVectorizer," *Dike J. Ilmu Multidisiplin*, vol. 1, pp. 37–43, 2023.

[29] T. K. K. D. Negeri, "Pedoman umum menghadapi PANDEMI COVID-19 bagi pemerintah daerah: pencegahan, pengendalian, diagnosis dan manajemen," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2020.

[30] R. Ramlan, N. Satyahadewi, and W. Andani, "Analisis Sentimen Pengguna Twitter Menggunakan Support Vector Machine Pada

Kasus Kenaikan Harga BBM," *Jambura J. Math.*, vol. 5, no. 2, pp. 431–445, 2023, doi: 10.34312/jjom.v5i2.20860.

[31] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022.

[32] J. Xu, Y. Zhang, and D. Miao, "Three-way confusion matrix for classification: A measure driven view," *Inf. Sci. (Ny).*, vol. 507, pp. 772–794, 2020.

[33] B. P. Salmon, W. Kleynhans, C. P. Schwegmann, and J. C. Olivier, "Proper comparison among methods using a confusion matrix," in *2015 IEEE International geoscience and remote sensing symposium (IGARSS)*, 2015, pp. 3057–3060.

[34] L. Kharb, D. Chahal, and Vagisha, "Forecasting Movie Rating Through Data Analytics," in *Data Science and Analytics: 5th International Conference on Recent Developments in Science, Engineering and Technology, REDSET 2019, Gurugram, India, November 15–16, 2019, Revised Selected Papers, Part II 5*, 2020, pp. 249–257.

[35] T. S. Gunawan, N. A. J. Abdullah, M. Kartiwi, and E. Ihsanto, "Social network analysis using python data mining," in *2020 8th international conference on cyber and IT service management (CITSM)*, 2020, pp. 1–6.

[36] S. Cao, Y. Zeng, S. Yang, and S. Cao, "Research on Python data visualization technology," in *Journal of Physics: Conference Series*, 2021, vol. 1757, no. 1, p. 12122.

[37] S. K. Mukhiya and U. Ahmed, *Hands-On Exploratory Data Analysis with Python: Perform EDA techniques to understand, summarize, and investigate your data*. Packt Publishing Ltd, 2020.

[38] M. Patel, "Exploratory Data Analysis and Sentiment Analysis on Brazilian E-Commerce Website." 2020.

[39] A. Setiawan, F. Setivani, and T. Mahatma, "Performance Comparison Of Decision Tree And Logistic Regression Methods For Classification Of SNP Genetic Data," *BAREKENG J. Ilmu Mat. dan Terap.*, vol. 18, no. 1, pp. 403–412, 2024.

[40] M. M. R. Khan, R. B. Arif, M. A. B. Siddique, and M. R. Oishe, "Study and observation of the variation of accuracies of KNN, SVM, LMNN, ENN algorithms on eleven different datasets from UCI machine learning repository," in *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, 2018, pp. 124–129.

[41] S. Rezvani, F. Pourpanah, C. P. Lim, and Q. M. J. Wu, "Methods for class-imbalanced learning with support vector machines: a review and an empirical evaluation," *Soft Comput.*, pp. 1–22, 2024.