Journal of Information Technology and Cyber Security 2(2) July 2024: 111-122 EISSN 2987-386X | PISSN 2987-3878
DOI: 10.30996/jitcs.12213

Dialect Classification of the Javanese Language Using the K-Nearest Neighbor

Brilliant Filby [1], Utomo Pujianto [2], Jehad A. H. Hammad [3], and Aji P. Wibawa [4,*]

[1,2] Department of Informatics Engineering, Universitas Negeri Malang, Indonesia
[3] Department of Computer Information Systems, Al-Quds Open University, Palestine
[4] Department of Electrical and Informatics Engineering, Universitas Negeri Malang, Indonesia
* Corresponding author: aji.prasetya.ft@um.ac.id

Received: 23 November 2024
Accepted: 12 January 2025
Revised: 09 January 2025
Available online: 20 January 2025

To cite this article: Filby, B., Pujianto, U., Hammad, J. A. H., & Wibawa, A. P. (2024). Dialect Classification of the Javanese Language Using the K-Nearest Neighbor. *Journal of Information Technology and Cyber Security*. 2(2), 111-122. https://doi.org/10.30996/jitcs.12213

Abstract

Indonesia is rich in ethnic and cultural diversity, each reflected in its unique linguistic characteristics. One way to preserve the Javanese language is by conducting research on its dialects. This study aims to classify three main dialects in Java Island—East Java, Central Java, and West Java—using text data from online sources. The classification process includes preprocessing (tokenizing, case folding, and word weighting), data balancing with the Synthetic Minority Oversampling Technique (SMOTE), and classification using the K-Nearest Neighbor (K-NN) algorithm. This study highlights the importance of dialect recognition in supporting the preservation of the Javanese language and the development of linguistic technology applications. Testing using 10-fold cross-validation showed the best performance at $k = 2$, with an accuracy of 94.05%, precision of 95.83%, and recall of 94.44%. These findings significantly support computational linguistics research and the preservation of regional languages.

Keywords: case folding, Javanese dialect, K-Nearest Neighbor, Natural Langugae Processing, Synthetic Minority Oversampling Technique, tokenizing.

1. Introduction

Language is the primary means of human communication, and it has variations in form and meaning depending on time, social group, and geographical location (Junaidi, Yani, & Rismayeti, 2016). In Indonesia, linguistic diversity is one of the characteristics, with 718 languages spread throughout the archipelago. Javanese has the most speakers, around 68.2 million people (Ethnologue, 2013). This language variation arises due to cultural, social, and geographical influences.

As one of the cultural heritages, Javanese has 12 dialect variations, including East Javanese, Central Javanese, and West Javanese dialects (Ethnologue, 2013). Several dialects are now facing the threat of extinction due to the shift to the use of Indonesian and foreign languages (Pamungkas & Hidayatullah, 2021). This decline affects local identity and the sustainability of regional culture. Therefore, research related to dialects is important to support language preservation.

Previous studies have examined the linguistic aspects of Javanese. For example, research by Ardhana (2018) classifies Javanese language levels such as ngoko, krama madya, and krama inggil using Multinomial Naïve Bayes (Ardhana, 2018). Meanwhile, Florensa (2021) utilized clustering-based K-Nearest Neighbor (K-NN) for Minang dialect classification, with the best accuracy reaching 88.3% (Irfa, Adiwijaya, & Mubarok, 2018). However, similar studies on Javanese dialects are still limited.

The K-NN algorithm is known as a reliable method for text classification. This algorithm works based on the distance between unknown and learning data, producing accurate predictions (Ardhana, 2018). Purnomo (2021) used K-NN to identify regional accents with an accuracy of up to 87.5%. In the context of this study, K-NN was chosen to classify the three main dialects of Javanese. This research approach also involves the Synthetic Minority Oversampling Technique (SMOTE) technique to handle data imbalance. This technique helps improve the accuracy of classification results by expanding the number of samples in mino-

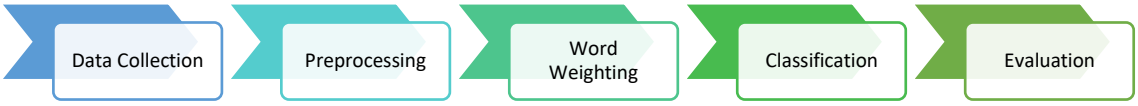

Fig. 1. Research flow diagram.

Table 1
Search results with keyword: East Java dialect.

Dialect	Search Results with Keyword
East Javanese	3,820
Osing	1,510
Tengger	2,570
Solo-Yogyakarta	3,570

Table 2
Search results with keyword: Central Java dialect.

Dialect	Search Results with Keyword
Pekalongan	3,880
Solo-Yogyakarta	3,570
Tegal	13,900
Banyumas	8,480
Wonosobo	299

Table 3
Search results with keyword: West Java dialect

Dialect	Search Results with Keyword
Pantai Utara	1,870
Cirebon	6,510
Ciamis	256

rity data groups (Palinoan, 2014). This is relevant due to the frequently uneven distribution of dialect data.

This study aims to fill the gap in the study of Javanese dialects by applying the K-NN algorithm and the SMOTE technique. In addition to contributing to preserving the Javanese language, this study is also expected to open up opportunities for other linguistic technology applications, such as speech recognition or Natural Language Processing (NLP) in the context of local culture.

2. Methods

The study has several stages: data collection, preprocessing, word weighting, classification, and evaluation, as shown in Fig. 1. This study began with data collection on online sites with various sources. Then, the data is processed in the preprocessing stage to be used as initial data for the next stage. After the data is ready, the word weighting process uses Term Frequency (TF). The classification process is carried out using the K-Nearest Neighbor (K-NN) method, and evaluation is performed using 10-fold cross-validation to assess the accuracy of the algorithm performance results.

2.1. Data Collection

The data collection method used in this study is collecting secondary data via the internet using Universal search. This study is called "blended/federated search results" (Sarwono, 2012). Sources and how to access them have a vital role in data quality (Kumar & Paul, 2016). So, to determine the accuracy of the data that has been collected, the author validates it with a linguist. Data retrieval is done by separating data based on dialect and retrieval source. Each dialect search is from one source, and several words and sentences are obtained. These words and sentences are entered into one Ms. Word document with the format (.docx) and saved with a name based on the dialect.

Furthermore, the number of words and sentences collected is calculated to ensure that the data balance between dialects can be adequately controlled. Because of the many dialects in Java, this study will only use dialects based on their distribution. These dialects are spread across East Java, West Java, and Central Java. To determine which dialects to use as a dataset, a survey was conducted by searching for "keywords" on Google to determine the most popular dialects in each region. The search was conducted using specific keywords for each dialect to calculate the search results for each dialect. For example, searching for the Tegal dialect using the keyword "Tegal dialect" produced 13,900 results.

Table 4
Search Results with Javanese Dialect Keyword.

Dialect	Word	Sentence	Total Words and Sentences	Documents
East Javanese	4,608	1,037	5,456	23
Tegal	5,625	1,103	6,726	20
Cirebon	4,608	1,037	5,208	10

Table 5
Preprocessing stages in several studies.

Researcher	Document Collection	Tokenize	Data Cleansing	Case Folding	Stopword Removal	Stemming	Term Frequency
Vijayarani, Ilamathi, & Nithya (2015)		✓	-	-	✓	✓	✓
Kannan & Gurusamy (2014)		✓	-	-	✓	✓	-
Kadhim (2018)	✓	✓	-	-	✓	✓	✓
Srividhya & Anitha (2010)		-	-	-	✓	✓	✓
Uysal & Gunal (2014)		✓	-	✓	✓	-	✓
Anandarajan, Hill, & Nolan (2019)		✓	✓	-	✓	✓	-
Denny & Spirling (2018)		✓	✓	✓	✓	✓	-
Sarkar (2019)		✓	✓	✓	✓	✓	-
Zong, Xia, & Zhang (2021)		✓	✓	-	✓	-	-
Elder, Miner, & Nisbet (2012)	✓	✓	✓	✓	✓	✓	✓

Furthermore, the search results for each dialect were recorded. From these search results, it is assumed that the more results found, the more popular the dialect is. Tables 1 to 3 show the most popular dialects based on the number of search results. The dialects selected for further data searches were the East Javanese dialect, with 3,820 search results; the Tegal dialect, with 13,900 results; and the Cirebon dialect, with 6,510 results. The total data collected was 17,390, consisting of 10 documents (5,208 absolute count) from the Cirebon dialect, 20 documents (6,726 absolute count) from the Tegal dialect, and 23 documents (5,456 absolute count) from the East Javanese dialect. The search results for the Javanese dialect can be seen in Table 4.

2.2. Preprocessing

One of the challenges in text mining or classification is converting unstructured and semi-structured text into structured files (Elder, Miner, & Nisbet, 2012). This process is known as preprocessing, which processes raw data into cleaner and more structured data so that it is ready to be used for further analysis processes (Ayub, 2007). The preprocessing stage must be carried out before proceeding to the following stages in data processing. Table 5 shows the search results for several studies that discuss the preprocessing stage. From the search, it can be concluded that there are seven preprocessing stages: document collection, data cleansing, tokenizing, case folding, stopword removal, stemming, and Term Frequency (TF). However, not all stages must be applied in every preprocessing process because each data has different processing needs. Of the several studies in Table 5, only two researchers include document collection as part of the preprocessing stage. In the study by Kadhim (2018), document collection was used for data labeling and division into two categories: training data and test data.

Meanwhile, in Elder, Miner, and Nisbet's (2012) study, the document collection stage was used to break down and summarize documents into several parts. From these two studies, it can be concluded that the document collection stage is used to process and prepare documents to become data ready to be processed. After that, the tokenization stage is carried out to break the text into separate words called tokens. Data cleaning eliminates noisy data, such as unnecessary symbols or characters. Furthermore, case folding is carried out to change all letters to lowercase. The stopword removal process aims to remove common words that are considered meaningless, such as the words "*di*", "*ke*", and "*dari*". The next stage is stemming, which removes word affixes and returns them to their basic word form. Finally, there is a word weighting stage, namely TF, which gives weight to the terms selected through the previous stages. This study used only five stages: document collection, tokenizing, cleaning data, case folding, and TF. Document collection is carried out to merge data. Case folding is used to match the letters in the data, thus producing

Algorithm 1. Data merging process.

```
Begin
Import libraries (readtext, dplyr)
    Create a list
    Read the dataset from the path and load it into the variable list_categories
    Create a dataframe with the variable df_final and create columns (File_Name, Content, and Category)
        For (category in list_categories) {
            Retrieve the dataset into the category and store it in category_path
            Read the dataset with the readtext library and load it into the variable df
            Insert file_name as File_Name, text as Content, Folder Name as Category
            Use the rbind function to append to the variable df_final (df_final = rbind(df_final, df))
        }
    Save the dataset using the save function with 'dataset_final.rda'
    Load the dataset with the load function and encode the data as .csv with fileEncoding = 'UTF-8'
End
```

a consistent format. Tokenize is used to break sentences into tokens, facilitating data analysis. The tokenization process includes the data cleaning stage, which removes irrelevant elements. Data cleaning and case folding are important in this study because programming languages are susceptible to differences in format, so changes to a uniform form are necessary. Finally, TF calculates how important a term is in a document.

Two stages not used in this study were stopword removal and stemming. These stages were not applied because it was assumed that all information contained in the dialect text was considered important. Therefore, the text must be recognized and cannot only be processed based on its root words. For example, using the word "*menyapu*" if only its root word, namely "*sapu*", is recognized can cause the word to be detected in another dialect. Thus, the words "*menyapu*" and "*sapu*" should not be counted as the same because their meanings in different dialect contexts can differ.

The collected data is still separated into documents stored in one folder. This folder has three subfolders: the East Javanese dialect folder, the Tegal dialect, and the Cirebon dialect. The available data files are 53 documents in docx format, so data merging is still needed. There are several ways to merge this data manually and using a programming language. Reading data from the file system is an essential skill possessed by almost all programming languages (Kumar & Paul, 2016). One language that can be used for this task is the R language, which provides various frameworks for accessing and performing various types of analysis on text. Algorithm 1 shows the steps used in the data merging process.

In Algorithm 1, two libraries are used, namely `readtext` and `dplyr`. The `readtext` library is used to read data in the dataset folder that has been created. The folder contains East Java, Tegal, and Cirebon folders: the `dplyr` library functions to create a data frame and the `rm list` function. Furthermore, the dataset is read from the path (dataset folder path address) and inserted into the `list_categories` variable. After the file is retrieved, a data frame with the `df_final` variable is created, and the columns are named, namely '`file name`', '`Content`' in this case is text, and '`category`'. In the iteration process for category in `list_categories`, the dataset for each category will be inserted into `category_path`.

Furthermore, category labels are created by reading the file using the `readtext` library and inserting it into the `df`. After that, the data from each category that has been read will be combined with the previous data using the `rbind (df final, df)` function. Dataset storage in Python uses the R language file format, so the data is loaded and encoded into .csv before being saved. UTF-8 encoding is important because Python cannot read the format without proper encoding, which can cause errors when the file is saved. The final result of the data merger produces one Excel document named 'final dataset' with 53 rows. During the merger process, each row is labeled according to the name of its original folder. There are 23 rows labeled 'East Javanese dialect,' 20 rows labeled 'Tegal dialect,' and 10 rows labeled 'Cirebon dialect.'

Furthermore, the case folding process is part of the text preprocessing stage to standardize the characters in the data. Case folding is changing all letters in the data to lowercase or capital letters. In this stage, characters from 'A' to 'Z' will be changed to lowercase ('a' to 'z'). Characters other than letters, such as punctuation and numbers, will be removed and considered delimiters (Jumeilah, 2017). A delimiter is a sequence of one or more characters used to determine the boundary of the separator. The output of this case, the folding process, will later be used as input for the tokenizing process. An example of the results of case-folding can be seen in Table 6.

The raw data from the dialect obtained is still in words and sentences. To make the analysis process

Table 6
Results of the preprocessing stage.

Process	Description	Input	Output
Case Folding	Converts all letters to lowercase.	*Aku kate nang Malang*	*aku kate nang malang*
Tokenize	Splits sentences into individual words or tokens.	*klambiku gurung garing*	*klambiku* *gurung* *garing*
Remove Punctuation	Removes symbols such as (!#$%&()*+,./:;<=>?@\[\\\]_{|}"~)	*kadang-kadang pancen kene mending ora eruh :(*	*kadang kadang pancen kene mending ora eruh*
Remove regular expression	Removes accented characters such as é, ÿ, ú	*celonoku teles masé*	*celonoku teles mas*
Remove Numbers	Removes numeric digits [0-9]	*tahun 2022 mbak iin wes umur 36*	*tahun mbak iin wes umur*

Algorithm 2. Data cleaning and tokenization process.

```
1  Begin
2  Import libraries (pandas, numpy, matplotlib.pyplot, re)
3      Read the .csv dataset (dataset_final.csv) using the library pd.read_csv as the variable df_FF
4      Create a variable cleaning = list ('0123456789éê!#$%&()*+,./:;<=>?@\[\\\]_{|}"~')
5      Create a variable cleaning = list ('éÿú')
6      Perform Tokenizing on the text
7  End
```

more manageable, the data needs to be segmented into words or tokens called tokenizing. The tokenizing stage is the cutting of the input string based on the words that make it up, or in other words, breaking sentences into words. Common strategies used in the tokenizing stage are cutting words at white space and removing punctuation characters. In addition, at this stage, characters and punctuation are also removed, which is part of the data-cleaning process. Data cleaning is a process to detect and repair (or delete) corrupt or inaccurate datasets, tables, and databases. This term refers to identifying incomplete, incorrect, imprecise, and irrelevant data, which will then be replaced, modified, or deleted (Pamungkas & Hidayatullah, 2021). The data cleaning and tokenization process will be explained further in Algorithm 2. In the data cleansing stage, some unnecessary punctuations or symbols will be removed, such as (!#$%&()*+,./:;<=>?@\[\\\]_{|}~). However, the hyphen (–) and single quotation marks (') are not removed because, in Javanese, there are many words that use hyphens, such as the word "*moro-moro*", and single quotation marks are often used in words such as "*opo'o*". Words like this must be detected intact with their punctuation so that they are not split into two words that may have different meanings.

Furthermore, other symbols in the form of regular expressions such as ú, ÿ, and é will be removed. Finally, the digit numbers [0-9] will also be removed because numbers are not relevant for the dialect classification that will be carried out. Examples of the results of the case folding, tokenizing, and data cleansing stages can be seen in Table 6.

The following preprocessing stage is the Synthetic Minority Oversampling Technique (SMOTE). Imbalanced data occurs if the number of objects in a data class exceeds others. This unbalanced data can affect the results obtained in model creation, so data balancing is needed. The SMOTE method is suggested by Chawla, Bowyer, Hall, and Kegelmeyer (2002) to handle class imbalance by avoiding the risk of overfitting often faced by random oversampling. This method adds the amount of minority class data to balance it with the majority class by using artificial samples generated by linearly interpolating randomly selected minority observations and one of its neighboring minority observations (Douzas, Bacao, & Last, 2018).

SMOTE consists of three main steps: clustering, filtering, and oversampling (Douzas, Bacao, & Last, 2018). In the clustering step, k-means clustering is used to group the minority class. This method aims to increase the class region by generating samples in natural clusters of the minority class. After the clustering step, a filtering step is performed to select the clusters to be oversampled and determine the number of samples to be generated in each cluster. This step aims to achieve a balanced distribution of samples in the minority class. Therefore, the filtering step allocates more generated samples to the minority clusters. Finally, SMOTE is applied to each selected cluster in the oversampling step to achieve the target ratio of

Table 7
Word frequency occurrence.

Word	Cirebon	Tegal	East Javanese	Total Words	Total Documents
Bebasan	1,806	0	0	1,806	2
Sing	75	376	466	917	48
Wong	40	190	238	468	37
Wis	25	220	212	457	37
Ana	58	198	188	444	38

minority and majority instances. In the data used in this study, there is an imbalance in the number of labels across the different dialects. The majority class consists of the East Javanese dialect, with a total of 23 labels, followed by the Tegal dialect with 20 labels, and the Cirebon dialect as the minority class with only 10 labels. Therefore, SMOTE was applied with a choice of 5 neighbors and a nominal change rate of 0.5, resulting in a balanced dataset with 23 labels for each of the East Javanese, Tegal, and Cirebon dialects. The ultimate goal of each resampling method is to improve the classification results. In other words, a resampling technique is considered successful if it enhances the prediction quality of the classifier used. Therefore, the effectiveness of the oversampling method can only be indirectly assessed by evaluating the classifier trained on the oversampled data.

2.3. Word Weighting

At the beginning of the classification process, it is necessary to select the documents to be classified and include them in the word weighting (Liao & Vemuri, 2002). One of the simplest weighting methods is TF. In this method, each term is considered to have a proportion of importance corresponding to the number of times it appears in the text, namely the frequency of term i\ in document\ j. Meanwhile, Inverse Document Frequency (IDF) measures the frequency of occurrence of terms across text documents. Terms that rarely appear in the entire document will have a higher IDF value than terms that appear frequently. Research combining TF and IDF to calculate term weights shows that combining the two produces better performance (Trstenjak, Mikac, & Donko, 2014). The TF-IDF method measures the relative frequency of a word in a particular document using the inverse proportion of the word across the entire document corpus. In calculating the TF-IDF value, this method involves two main elements: TF (frequency of occurrence of term i in document j) and IDF (inverse frequency of documents containing term i). The TF-IDF calculation formula can be seen in Eq. (1).

$$a_{ij} = tf_{ij} idf_i = tf_{ij} \times log_2\left(\frac{N}{df_i}\right) \quad (1)$$

in Eq. (1), a_{ij} represents the weight of term i in document j. Furthermore, N refers to the total number of documents in the data collection. The value of tf_{ij} is the frequency of occurrence of term i in document j. As for df_i, this parameter refers to the number of documents in the collection containing the term i.

For example, in the analysis of the frequency of occurrence of words in the Javanese dialect, it can be seen in Table 7 that the most frequently occurring word in the Cirebon dialect is "*bebasan*", which appears 1,806 times in 2 documents. Meanwhile, in the Tegal and East Javanese dialects, the most frequently occurring word is "*sing*", which is spread across 40 documents with a frequency of occurrence of 376 times in the Tegal dialect and 466 times in the East Javanese dialect.

2.4. Classification using k-Nearest Neighbor (KNN)

K-Nearest Neighbor (K-NN) is an algorithm used to classify data based on its k nearest neighbors, with k referring to the number of neighbors considered in the classification process (Isnain, Supriyanto, & Kharisma, 2021). K-NN is included in the category of supervised learning algorithms, which aim to find patterns in data by connecting new data with existing patterns. In the K-NN algorithm, distance measurement plays an important role in determining the level of similarity or regularity between data and items (Mughnyanti, 2020). Research conducted by Wahyono, Trisna, Sariwening, Fajar, & Wijayanto (2020), which compared distance calculations in K-NN for textual data classification, showed that Euclidean distance provides the best accuracy at most k sizes. The formula for calculating Euclidean distance can be seen in Eq. (2).

$$d(x,y) = \sqrt{\Sigma_{r=1}^{N}(a_{rx} - a_{ry})^2} \quad (2)$$

Eq. (2) defines $d(x,y)$, epresenting the distance between two documents, x and y. In this context, N refers to the total number of unique words in the document collection. The parameters a_{rx} dan a_{ry} denote the term r's weight in documents x and y, respectively.

In addition, the value of k in the K-NN algorithm is a factor that determines the number of documents

Table 8
Some studies on k modification.

Researcher	Best k	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
Briliani, Irawan, & Setianingsih (2019)	3	98	98	98	98.13
Asiyah (2015)	2	90.27	83.97	87	83.97
Nurjanah, Perdana, & Fauzi (2017)	3	72.28	100	83.91	80.83
Irfan (2020)	5	61.66	56	58.66	81.81

Table 9
Testing scheme for various k values.

k	Accuracy (%)	Precision (%)	Recall (%)
2	**94.05**	**95.83**	**94.44**
3	91.43	93.61	91.67
4	91.19	94.17	92.22
5	90.00	93.83	90.56
6	90.00	93.83	90.56
7	87.14	90.83	88.33
8	87.14	92.17	88.33
9	85.71	88.61	86.67

from the collection that are closest to the document to be selected. Determining the optimal value of k is highly dependent on the characteristics of the data itself. In general, a higher value of k can reduce the effect of noise on classification but can make the boundaries between each classification less clear. To overcome this problem, the value of k can be modified in each class (Khamar, 2013). Several studies related to modifying the value of k can be seen in Table 8.

From Table 8, several studies show that the higher the value of k, the lower the accuracy, although there are certain conditions where a particular value of k increases accuracy. Increasing the k in each distance calculation tends to decrease accuracy because the more k values are used, the more data is not classified correctly. Therefore, in this study, only testing was carried out on the k variable with a smaller value, namely $k = 2$ to $k = 9$.

2.5. Evaluation

The cross-validation method is used to test the K-NN algorithm. Cross-validation is a statistical technique employed to evaluate and compare learning algorithms by partitioning the data into two sets: training data and testing data. The data, divided into these two parts, are alternately used for training and testing. A commonly used technique in this process is K-Fold Cross Validation. In K-Fold Cross Validation, the complete dataset is randomly divided into 'k' subsets of approximately equal size and mutually exclusive. The model in classification is then trained and tested 'k' times, with each training run performed on all folds except one, which is left out for testing. In this study, 10-fold cross-validation is used. The evaluation of classification results includes three key metrics: accuracy, precision, and recall.

3. Results and Discussion

This chapter discusses the stages of testing and analyzing the results of implementing the K-NN algorithm for classifying Javanese dialects. The study was conducted using one merged document consisting of three labels, namely the East Java label, the Tegal label, and the Cirebon label. The previously available data was balanced using the SMOTE oversampling technique with a value of $k = 6$, resulting in a total of 69 labels evenly divided into three for each label. Before testing the K-NN algorithm, the data was divided into training and test data using the 10-fold cross-validation method, where 90% of the data was used as training data and 10% as test data.

The K-NN algorithm testing was carried out with a k value testing scenario, namely, k values varying from 2 to 9. This test aims to determine the optimal k value in the classification process using the K-NN algorithm, especially in determining the system's accuracy results. Each k value becomes a testing parameter, so its effect on system accuracy can be analyzed. The results of testing the effect of the k value on system accuracy are presented in Table 9. At the same time, a visual summary is shown in Fig. 2. Based on the test results, several k values showed the same performance, such as $k = 5$ and $k = 6$ and $k = 7$, and $k = 8$. This is due to the mechanism of the K-NN algorithm, where the k value represents the number of closest data used in the classification. If the data labels on the k nearest neighbors vary, then the classification will be determined based on the majority label. Therefore, in practice, the k value is generally an odd number.

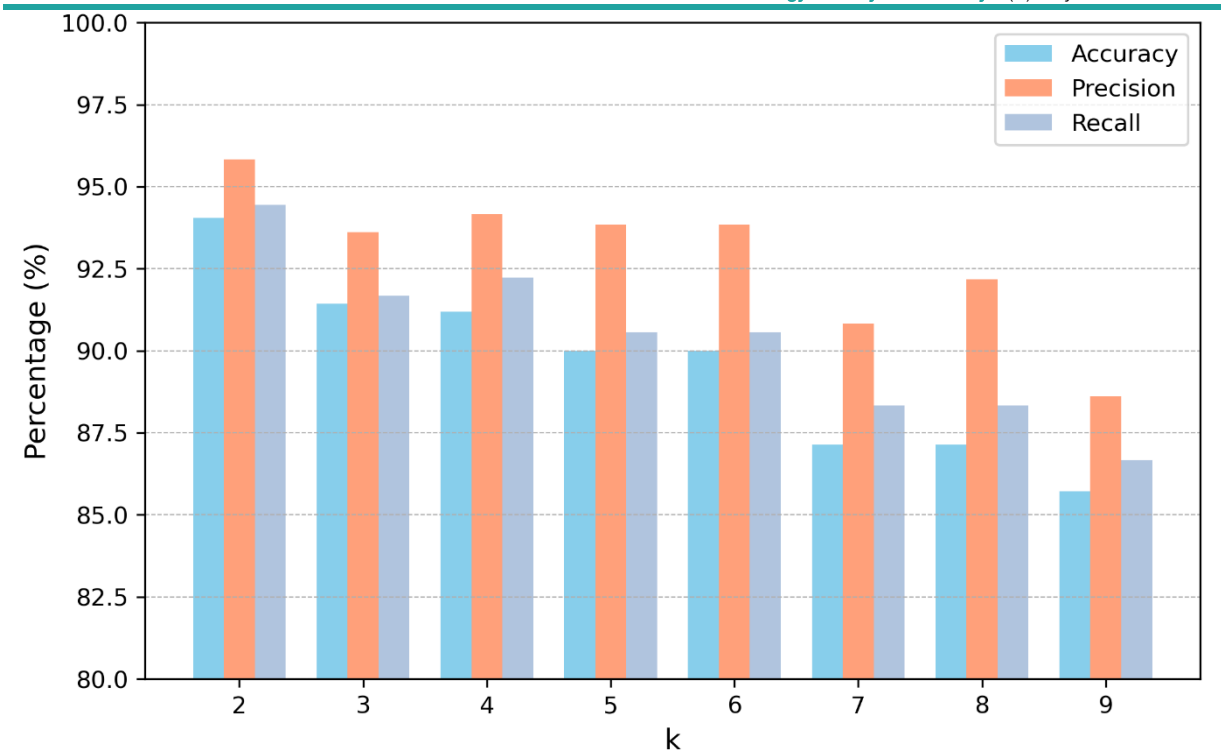

Fig. 2. Results of testing based on k values.

The results of testing the effect of the k value on system accuracy are presented in Table 9. At the same time, a visual summary is shown in Fig. 2. Based on the test results, several k values showed the same performance, such as $k = 5$ and $k = 6$ and $k = 7$, and $k = 8$. This is due to the mechanism of the K-NN algorithm, where the k value represents the number of closest data used in the classification. If the data labels on the k nearest neighbors vary, then the classification will be determined based on the majority label. Therefore, in practice, the k value is generally an odd number. The test results show that the best classification performance is obtained at a value of $k = 2$, with an accuracy of 94.05%, a precision of 95.83%, and a recall of 94.44%. Visualization of the confusion matrix for each label classified at $k = 2$ is shown in Fig. 3.

From the 69 labels, 65 were predicted correctly, and 4 were mispredicted. The details are as follows: 21 labels were predicted correctly for the Cirebon dialect, while 2 labels were incorrectly predicted as the East Javanese dialect. The tegal dialect has entirely correct predictions. Meanwhile, 21 labels were predicted correctly for the East Javanese dialect, while two were incorrectly predicted as the Cirebon dialect. Calculation for each label is done using evaluation metrics. Precision value is calculated by dividing the number of relevant documents by the number of all documents found.

In contrast, recall value is calculated by dividing the number of relevant documents by the number of all documents in the data. Meanwhile, the accuracy value is calculated by dividing the number of correct predictions by the number of all predictions and multiplying by 100%. The results of calculating accuracy, precision, and recall for each dialect are presented in Table 10.

The accuracy for Cirebon and East Javanese dialects of 91.30% is caused by two documents that are mispredicted in both dialects. Meanwhile, the Tegal dialect obtained 100% accuracy because all documents were successfully predicted correctly as the Tegal dialect. Table 11 shows data with incorrect label predictions.

The incorrect prediction of the Cirebon class as East Javanese refers to a document containing the text "*uwoh srikayo di paih...*". In this document, the confidence value for the Tegal class is 0.450, while the East Javanese class's is 0.550. Therefore, the document is predicted to be the East Javanese dialect because the East Javanese class has a higher confidence value. Likewise, this document is predicted to be the Cirebon dialect in the East Javanese dialect containing the text "*ngalam arudam arodam...*". This happens because the confidence value for the East Javanese class of 0.451 is smaller than that for the Cirebon dialect, which reaches 0.549. Thus, the class prediction falls on the Cirebon dialect, according to the class with the highest confidence value in the text.

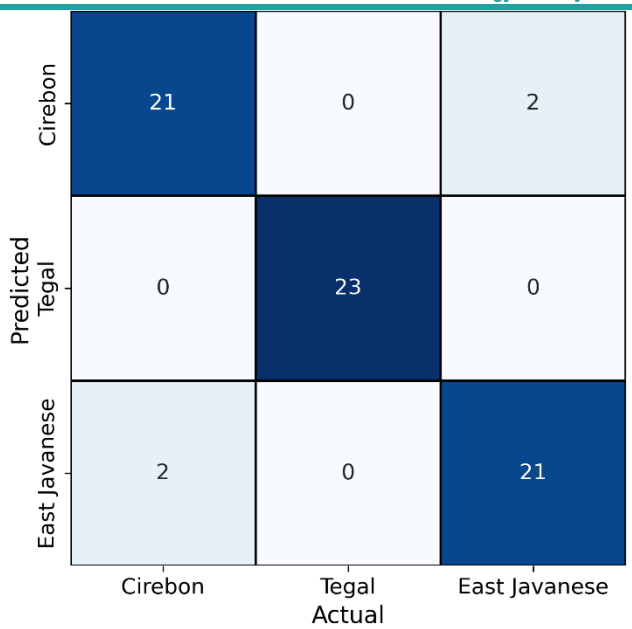

Fig. 3. Confusion matrix from the testing results at $k = 2$.

Table 10
Accuracy, precision, and recall values for each class.

Dialect	Accuracy	Precision	Recall
Cirebon	91.30	91.30	91.30
Tegal	100	100	100
East Javanese	91.30	91.30	91.30

Table 11
Incorrect class prediction.

Class	Prediction (class)	Confidence (Tegal)	Confidence (Cirebon)	Confidence (East Javanese)	Text
Cirebon	East Javanese	0.450	0	0.550	*uwoh srikayo di paih…*
Cirebon	East Javanese	0	0.450	0.550	*aang abah abal aban...*
East Javanese	Cirebon	0	0.528	0.472	*ngalam arudam arodam…*
East Javanese	Cirebon	0	0.549	0.451	*a iku ojo ngono ta yok…*

The findings of this study highlight the effectiveness of the K-NN algorithm in classifying dialect texts, primarily when supported by proper preprocessing and data balancing using SMOTE. A smaller k value provides optimal results because it narrows the scope of relevant neighboring data and increases sensitivity to noise in the dataset. Therefore, the selection of the k value must consider the characteristics of the dataset as a whole to achieve more accurate results.

Although the model's accuracy is relatively high, several weaknesses were identified, such as bias in the distribution of prediction results. The Cirebon dialect, for example, is often classified as East Javanese, indicating the need for more specific feature adjustments to capture unique differences between dialects. In addition, although the SMOTE technique effectively balances data, using less representative synthetic data can introduce new biases if not adequately supervised.

The results of this study have important implications for preserving the Javanese language, especially in supporting the development of linguistic technology-based applications. Integration of the results of this study with other methods, such as neural network-based embedding, can provide more in-depth and high-precision results in the future.

4. Conclusions

This study successfully classifies three main dialects of Javanese, namely East Javanese, Central Javanese, and West Javanese, using the K-NN algorithm with the highest accuracy of 94.05% at a value of $k = 2$. The K-NN algorithm has been proven effective in capturing differences in linguistic patterns between dialects, especially with the support of the SMOTE technique, which balances uneven data. In addition, the

preprocessing process involving tokenizing, case folding, and word weighting using TF also plays an important role in improving model performance.

A significant contribution of this study lies in its support for preserving the Javanese language through a linguistic technology approach. The results of this study can also be a basis for further development in the field of dialect-based Natural Language Processing (NLP), such as speech recognition applications and text analysis. However, some limitations, such as data bias from online sources and data distribution that can still be improved, must be considered in further research. Therefore, efforts to expand the scope of data by involving authentic sources and exploring other classification algorithms, such as SVM or deep learning, can improve the quality and impact of this study.

5. CRediT Authorship Contribution Statement

Brilliant Filby: Formal Analysis, Investigation, Software, Visualization, and Writing – Original Draft. **Utomo Pujianto:** Formal Analysis, Investigation, Methodology, Resources, Writing – Original Draft, and Writing – review & editing. **Jehad A. H. Hammad:** Formal Analysis, Validation and Writing – Review & Editing. **Aji P. Wibawa:** Conceptualization, Data curation, Formal Analysis, Investigation, Methodology, Project administration, Supervision, and Writing – review & editing.

6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7. Data Availability

Data will be made available on request.

8. References

Anandarajan, M., Hill, C., & Nolan, T. (2019). *Practical Text Analytics: Maximizing the Value of Text Data.* Springer Cham. doi:https://doi.org/10.1007/978-3-319-95663-3

Ardhana, A. P. (2018). *Klasifikasi Tingkatan Bahasa pada Artikel Berbahasa Jawa dengan Metode Multinomial Naïve Bayes.* Surakarta, Indonesia: Universitas Sebelas Maret. Retrieved from https://digilib.uns.ac.id/dokumen/detail/58424/

Asiyah, S. N. (2016). *Online News Classification Using Support Vector Machine and K-Nearest Neighbor.* Surabaya, Indonesia: Institut Teknologi Sepuluh Nopember. Retrieved from https://repository.its.ac.id/62883/1/1314105016-Undergradute%20Thesis.pdf

Ayub, M. (2007). Proses Data Mining dalam Sistem Pembelajaran Berbantuan Komputer. *Jurnal Sistem Informasi, 2*(1), 21-30. Retrieved from https://www.researchgate.net/profile/Mewati-Ayub/publication/237692809_Proses_Data_Mining_dalam_Sistem_Pembelajaran_Berbantuan_Komputer/links/5aeefe5c0f7e9b01d3e2bd70/Proses-Data-Mining-dalam-Sistem-Pembelajaran-Berbantuan-Komputer.pdf?_tp=eyJjb250ZXh0Ijp

Briliani, A., Irawan, B., & Setianingsih, C. (2019). Hate Speech Detection in Indonesian Language on Instagram Comment Section Using K-Nearest Neighbor Classification Method. *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)* (pp. 98-104). Bali, Indonesia: IEEE. doi:https://doi.org/10.1109/IoTaIS47347.2019.8980398

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research, 16*, 321–357. doi:https://doi.org/10.1613/jair.953

Denny, M. J., & Spirling, A. (2018). Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It. *Political Analysis, 26*(2), 168-189. doi:https://doi.org/10.1017/pan.2017.44

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences, 465*. doi:https://doi.org/10.1016/j.ins.2018.06.056

Elder, J., Miner, G., & Nisbet, B. (2012). *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications.* Middlesex County, United States: Academic Press. Retrieved from https://books.google.co.id/books?id=-B6amxqygTMC&dq=G.+Miner,+Practical+Text+Mining+and+Statistical+Analysis+for+Non-Structured+Text+Data+Applications.+Elsevier+Science,+2012.&lr=&hl=id&source=gbs_navlinks_s

Ethnologue. (2013, Feb 28). *Methodology*. Retrieved from Ethnologue: https://www.ethnologue.com/methodology/

Florensa, R. (2021). *Peningkatan Kecepatan Pencarian K-Nearest Neighbour Berbasis Clustering pada Dialek Bahasa Minang.* Yogyakarta, Indonesia: Universitas Gadjah Mada. Retrieved from https://etd.repository.ugm.ac.id/penelitian/detail/205771

Irfa, A. A., Adiwijaya, A., & Mubarok, M. S. (2018). Klasifikasi Topik Berita Berbahasa Indonesia Menggunakan k-Nearest Neighbor. *Proceedings of Engineering. 5*, pp. 3631-3640. Bandung, Indonesia: Universitas Telkom. Retrieved from https://core.ac.uk/download/pdf/299923375.pdf

Irfan, R. (2020). *Analisis Perbandingan Algoritma K-Nearest Neighbor dan Support Vector Machine pada Pengklasifikasian Hadits Shahih Muslim.* Jakarta, Indonesia: Universitas Islam Negeri Syarif Hidayatullah. Retrieved from https://repository.uinjkt.ac.id/dspace/bitstream/123456789/55999/1/RENALDY%20IRFAN-FST.pdf

Isnain, A. R., Supriyanto, J., & Kharisma, M. P. (2021). Implementation of K-Nearest Neighbor (K-NN) Algorithm For Public Sentiment Analysis of Online Learning. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 15*(2), 121-130. Retrieved from https://jurnal.ugm.ac.id/ijccs/issue/view/4602

Jumeilah, F. S. (2017). Penerapan Support Vector Machine (SVM) untuk Pengkategorian Penelitian. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 1*(1), 19 - 25. doi:https://doi.org/10.29207/resti.v1i1.11

Junaidi, J., Yani, J., & Rismayeti, R. (2016). Variasi Inovasi Leksikal Bahasa Melayu Riau di Kecamatan Pulau Merbau. *Jurnal Pustaka Budaya, 3*(1), 1-16. Retrieved from https://journal.unilak.ac.id/index.php/pb/article/view/582

Kadhim, A. I. (2018). An Evaluation of Preprocessing Techniques for Text Classification. *International Journal of Computer Science and Information Security (IJCSIS), 16*(6). Retrieved from https://www.researchgate.net/profile/Ammar-Kadhim-4/publication/329339664_An_Evaluation_of_Preprocessing_Techniques_for_Text_Classification/links/5c1b6aa6a6fdccfc705ae648/An-Evaluation-of-Preprocessing-Techniques-for-Text-Classification.pdf?_tp=eyJjb250ZX

Kannan, S., & Gurusamy, V. (2014). Preprocessing Techniques for Text Mining. *International Journal of Computer Science & Communication Networks, 5*(1), 7-16. Retrieved from https://www.researchgate.net/profile/Vairaprakash-Gurusamy/publication/273127322_Preprocessing_Techniques_for_Text_Mining/links/54f8319e0cf210398e949292/Preprocessing-Techniques-for-Text-Mining.pdf

Khamar, K. (2013). Short Text Classification Using kNN Based on Distance Function. *International Journal of Advanced Research in Computer and Communication Engineering, 2*(4), 1916-1919. Retrieved from https://www.academia.edu/download/38502879/knn.pdf

Kumar, A., & Paul, A. (2016). *Mastering Text Mining with R.* Birmingham, UK: Packt Publishing. Retrieved from https://www.oreilly.com/library/view/mastering-text-mining/9781783551811/

Liao, Y., & Vemuri, V. (2002). Use of K-Nearest Neighbor classifier for intrusion detection. *Computers & Security, 21*(5), 439-448. doi:https://doi.org/10.1016/S0167-4048(02)00514-X

Mughnyanti, M. (2020). *Analisis penggunaan Manhattan distance dan euclidean distance pada algoritma x-means dalam pengelompokan data.* Medan, Indonesia: Universitas Sumatera Utara. Retrieved from https://digilib.usu.ac.id/en/detail.php?ib=201023104920853&i=

Nurjanah, W. E., Perdana, R. S., & Fauzi, M. A. (2017). Analisis Sentimen Terhadap Tayangan Televisi Berdasarkan Opini Masyarakat pada Media Sosial Twitter menggunakan Metode K-Nearest Neighbor dan Pembobotan Jumlah Retweet. *JPTIIK (Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer), 1*(12), 1750–1757. Retrieved from https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/631

Palinoan, V. W. (2014). *Sistem Klasifikasi Dokumen Bahasa Jawa dengan Metode K-Nearest Neighbour.* Sleman, Indonesia: Universitas Sanata Dharma. Retrieved from https://repository.usd.ac.id/4346/

Pamungkas, R. D., & Hidayatullah, A. F. (2021). Tinjauan Literatur : Identifikasi Dialek Dengan Deep Learning. *Automata. 2.* Yogyakarta, Indonesia: Universitas Islam Indonesia. Retrieved from https://journal.uii.ac.id/AUTOMATA/article/view/17390

Purnomo, G. W. (2021). *Identifikasi Asal Daerah Berdasarkan Logat Manusia dengan Metode Linear Predictive Coding (LPC) dan K-Nearest Neighbor (K-NN).* Bandung, Indonesia: Universitas Telkom. Retrieved from https://openlibrary.telkomuniversity.ac.id/home/catalog/id/175067/slug/identifikasi-

asal-daerah-berdasarkan-logat-manusia-dengan-metode-linear-predictive-coding-lpc-dan-k-nearest-neighbor-k-nn-.html
Sarkar, D. (2019). *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing.* Apress Berkeley. doi:https://doi.org/10.1007/978-1-4842-4354-1
Sarwono, J. (2012). *Metode Riset Online: Teori, Praktik, dan Pembuatan Apliaksi (Menggunakan HTML, PHP, dan CSS).* Jakarta, Indonesia: Elex Media Komputindo. Retrieved from https://books.google.co.id/books?id=dttMDwAAQBAJ&hl=id&source=gbs_navlinks_s
Srividhya, V., & Anitha, R. (2010). Evaluating Preprocessing Techniques in Text Categorization. *International Journal of Computer Science and Application, 47*(11), 49-51. Retrieved from http://sinhgad.edu/ijcsa-2012/pdfpapers/1_11.pdf
Trstenjak, B., Mikac, S., & Donko, D. (2014). KNN with TF-IDF based Framework for Text Categorization. *Procedia Engineering*, *69*, pp. 1356-1364. doi:https://doi.org/10.1016/j.proeng.2014.03.129
Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management, 50*(1), 104-112. doi:https://doi.org/10.1016/j.ipm.2013.08.006
Vijayarani, S., Ilamathi, J., & Nithya, N. (2015). Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science & Communication Networks, 5*(1), 7-16. Retrieved from https://www.ttcenter.ir/ArticleFiles/ENARTICLE/3783.pdf
Wahyono, W., Trisna, I. N., Sariwening, S. L., Fajar, M., & Wijayanto, D. (2020). Comparison of distance measurement on k-nearest neighbour in textual data classification. *Jurnal Teknologi dan Sistem Komputer, 8*(1), 54-58. doi:https://doi.org/10.14710/jtsiskom.8.1.2020.54-58
Zong, C., Xia, R., & Zhang, J. (2021). *Text Data Mining.* Singapore: Springer. doi:https://doi.org/10.1007/978-981-16-0100-2