JURNAL INFORMATIKA, Vol.11 No.2 October 2024, Page 73-79
ISSN (Print) 2355-6579 | ISSN (Online) 2528-2247
DOI: https://doi.org/10.31294/inf.v11i2.21107

Analysis of FastText with Support Vector Machine for Hate Speech Classification on Twitter Social Media

Nabila Nuraini[1*], Naufal Azmi Verdikha[2], Asslia Johar Latipah[3]

[1,2,3] Informatics Engineering, Faculty of Science and Technology, Muhammadiyah University East Kalimantan
Jl. Ir. H. Juanda No.15, Sidodadi, Kec. Samarinda Ulu, Samarinda, East Kalimantan

Correspondence e-mail: 1911102441033@umkt.ac.id

Submission:	Revision:	Acceptance:	Available Online:
18-01-2024	01-06-2024	29-07-2024	23-08-2024

Abstract

Hate speech refers to sentences or words aimed at degrading or insulting individuals, groups, or communities based on factors such as ethnicity, religion, race, or intergroup differences. Hate speech has harmful impacts, including increasing social tension, triggering conflicts, and inciting violence. Therefore, automatic detection of hate speech is crucial to maintaining societal harmony and preventing conflict escalation. This study employs Natural Language Processing (NLP) techniques using FastText feature extraction and the SVM algorithm for text classification. The evaluation is conducted using the F1 Score as the performance metric. The data is split using the 10-fold Cross Validation method, and experiments are carried out with four SVM kernels: RBF, Linear, Polynomial, and Sigmoid. The research findings demonstrate that the combination of FastText and SVM methods is effective in hate speech classification. By adopting FastText parameters from previous studies and involving four SVM kernels, this study achieves a satisfactory average F1 Score. The Polynomial kernel shows the best performance with an F1 Score of 0.813, followed by the Linear kernel with 0.809, the RBF kernel with 0.808, and the Sigmoid kernel with 0.805, indicating that the differences in F1 Score results are not significant. These findings underscore the importance of selecting the appropriate kernel in SVM to improve the accuracy of hate speech detection.

Keywords: Hate Speech, FastText Feature Extraction & Support Vector Machine (SVM)

1. Introduction

Hate speech is an important issue that is attracting the attention of many parties, especially in today's digital era. Cases of hate speech are frequently encountered on social media, particularly on platforms like Twitter. The use of social media often triggers the emergence of opinions that qualify as hate speech and defamation. Hate speech consists of sentences or words that demean or insult individuals, groups, or specific communities, containing elements of ethnicity, religion, race, and intergroup differences (Antariksa et al., 2019).

Such actions can have damaging negative impacts at both the individual and social levels. Individuals who fall victim to hate speech often experience psychological trauma, stress, and depression, which can diminish their quality of life. At the social level, hate speech has the potential to incite intergroup conflicts, increase social polarization, and reinforce negative stereotypes, all of which contribute to societal division and threaten social cohesion. In extreme cases, hate speech can provoke physical violence and criminal acts, disrupting public safety and order.

Natural Language Processing (NLP) studies the interaction between human natural language and machines to understand the meaning and analyze the expressions conveyed through words or text. In language processing, NLP employs techniques that represent words in vector form, known as word embedding, such as Word2Vec, GloVe, and FastText, which are implemented using machine learning models (Kedia & Rasu, 2020). One of the machine learning models that can be used for classification is the Support Vector Machine (SVM), which is utilized for text classification, object recognition, and text recognition (Adhari et al., 2021).

Furthermore, a study related to NLP conducted by Amalia et al. (2020) compared TF-IDF and FastText in text classification using the F1 Score evaluation metric. In the initial experiment, the FastText model used default parameters and yielded suboptimal performance. Consequently, the researchers repeated the experiment to find the best hyperparameters to optimize the evaluation metric. The hyperparameters used were a learning rate of 1, epoch of 10, and NGram of 2. From this comparison, the FastText model showed an improvement in the F1 Score from 0.74 to 0.97, with a processing time of 0.0459109999 seconds, which is faster than the TF-IDF model's time of 1.478 seconds and an F1 Score of 0.97. Therefore, the parameters of the FastText model significantly affect the F1 Score evaluation results, enhancing the model's performance.

A comparative study combining several other methods was conducted by Baskoro et al. (2021) on Software Requirements Specification (SRS) data using two datasets, PROMISE and SecReq. The comparison involved methods such as SVM, SVM-FastText, Convolutional Neural Network (CNN), and CNN-FastText. The results showed that SVM-FastText achieved an F1 Score of 0.90 on both datasets, while CNN-FastText obtained an F1 Score of 0.84 on PROMISE and 0.74 on SecReq. This discrepancy is due to the fact that CNN-FastText incorporates hidden layers, necessitating suitable parameters to effectively combine the two methods.

Furthermore, research by Boucherit & Abainia (2022) on offensive language data employed several methods, including SVM, Multinomial NB, Gaussian NB, CNN, BiLSTM, and FastText. In both binary and multi-label classification, SVM and Multinomial NB outperformed the other methods, while FastText provided the highest results for feature extraction compared to other methods. Related to classification with SVM in text classification, Adhari et al. (2021) designed a website to detect Twitter posts that contain threats using Support Vector Machine (SVM). The evaluation used four parameters: accuracy, precision, recall, and F1 Score. The highest evaluation result from six tests showed the best accuracy value of 73%.

Thus, the research has yet to explore the combination of FastText and SVM methods for classification using secondary data from the study by Ibrohim & Budi (2019), titled “Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter.” This study detects hate speech and abusive language with multi-label classification, where the dataset is categorized into multiple classes. The classes or labels include 12 categories based on the type of speech: Hate Speech, Abusive Language, Individual Hate Speech, Group Hate Speech, Religious Hate Speech, Gender Hate Speech, Racial Hate Speech, Physical Hate Speech, Other Hate Speech, Weak Hate Speech, Moderate Hate Speech, and Strong Hate Speech. The dataset consists of 5,561 tweets classified as hate speech, 5,043 tweets as abusive, and 7,604 tweets as non-hate speech. Among these categories, there are classes with similar meanings, such as religious hate speech, gender hate speech, racial hate speech, and physical hate speech.

This study conducts experiments and analyzes the results of hate speech classification using data from the research by Ibrohim & Budi (2019). FastText is employed for feature extraction using the parameters outlined in the study by Amalia et al. (2020). Subsequently, the classification model is developed using the SVM algorithm, with performance evaluated using the F1 Score metric.

2. Research Methods

In this study, FastText will be used for Natural Language Processing (NLP) and Support Vector Machine (SVM) for classification. Figure 1 below illustrates the research flow to be undertaken.

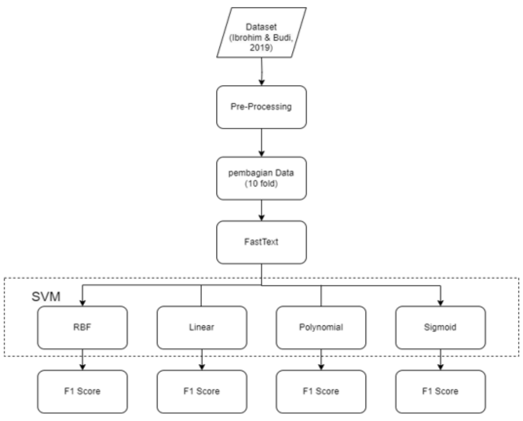


Image 1. Research flow

The data used in this study is sourced from Ibrohim & Budi (2019), as detailed in the Introduction section. This dataset is open source and can be obtained from the GitHub site.

2.1 Data PreProcessing

At this stage, several data cleaning steps will be performed on the dataset. The following are the preprocessing steps:

a. *Insert Column ID*
This step utilizes the Pandas library (McKinney, 2010) to add an ID column to the dataset. This addition helps in identifying, tracking, and processing the data more efficiently and accurately.

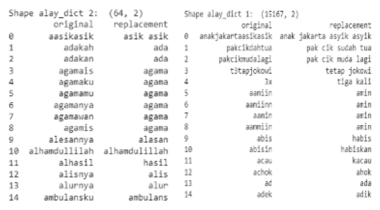
b. *Lower Case*
This step uses the Regular Expression library (Python Software Foundation, 2023) to convert the text in the tweet column to lowercase. This transformation helps avoid discrepancies in characters that should be identical but differ due to capitalization differences.

c. *Remove Attribute Tweet*
This step employs the Regular Expression (re) library (Python Software Foundation, 2023) to remove specific attributes from the

tweets, such as URLs, retweet symbols, usernames, emojis, and repeated characters.

d. *Non Alphanumeric*
This step uses the Regular Expression (re) library (Python Software Foundation, 2023) to remove all alphanumeric characters and any unwanted characters, such as punctuation, symbols, or other special characters.

e. *Spell Checker*
For spell checking, two slang dictionaries are used, as shown in Figure 2 below. These dictionaries are employed to correct spelling in the text to align with the standard Indonesian language as defined by the KBBI (Kamus Besar Bahasa Indonesia). The "alay_dict 2" dictionary contains predominantly regional language words, so corrections are informed by native speakers of the respective ethnic groups. This dictionary is imported using the Pandas library (McKinney, 2010).



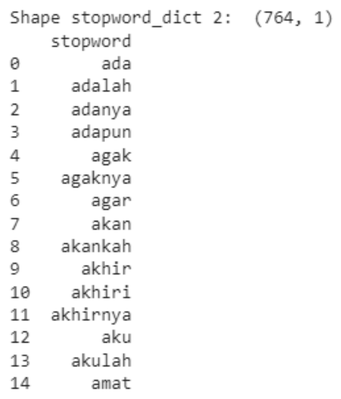
```
Shape alay_dict 2:  (64, 2)
          original   replacement
0        aasikasik     asik asik
1          adakah           ada
2          adakan           ada
3          agamais         agama
4          agamaku         agama
5          agamamu         agama
6         agamanya         agama
7         agamawan         agama
8           agamis         agama
9        alesannya        alasan
10   alhamdullilah  alhamdulillah
11         alhasil         hasil
12         alisnya          alis
13         alurnya          alur
14      ambulansku      ambulans
```

```
Shape alay_dict 1:  (15167, 2)
                  original                  replacement
0    anakjakartaasikasik  anak jakarta asyik asyik
1          pakcikdahtua         pak cik sudah tua
2        pakcikmudalagi         pak cik muda lagi
3           t3tapjokowi              tetap jokowi
4                    3x                 tiga kali
5                aamiin                      amin
6               aamiinn                      amin
7                 aamin                      amin
8               aammiin                      amin
9                  abis                     habis
10               abisin                  habiskan
11                 acau                     kacau
12                achok                      ahok
13                   ad                       ada
14                 adek                      adik
```

Image 2. Alay Dictionary

f. *Stemming*
The stemming step involves converting words to their base forms using the Sastrawi library (Cahyono, 2023). For instance, the word "mencegah" is reduced to the base form "cegah," "membantu" to "bantu," and "berenang" to "renang.".

g. *Stopword*
In the stopword removal step, common words that do not provide significant information, such as “ini,” “itu,” and “adalah,” are removed, as shown in Figure 3. This process uses a stopword list in Indonesian, which is imported using the Pandas library (McKinney, 2010).



```
Shape stopword_dict 2:  (764, 1)
    stopword
0        ada
1     adalah
2     adanya
3     adapun
4       agak
5    agaknya
6       agar
7       akan
8    akankah
9      akhir
10    akhiri
11  akhirnya
12       aku
13    akulah
14      amat
```

Image 3. Indonesian Stopwords

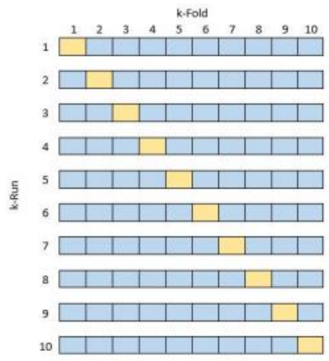
h. Word Visualization
In the word visualization step, the `collections` module is used to count the frequency of occurrences of elements within the data. Subsequently, the `WordCloud` module generates a word cloud image based on the frequency data.

i. *Remove Empty Data*
In this section, empty data, data consisting only of spaces, and data containing only a single word are removed.

2.2 Cross Validation

Cross Validation addresses this issue by dividing the data into several subsets called folds. In each iteration, one fold is used as the test set, while the remaining folds are used as the training set. This ensures that each fold is used as a test set at various points during the cross-validation process (Peryanto et al., 2020).

This step involves splitting the data using cross-validation with 10 folds, employing the KFold module from the Sklearn library (Pedregosa et al., 2011). In the first iteration, the first fold is used as the test set, and the remaining 9 folds are used as the training set. The process is illustrated in Figure 4 below for the 10-fold cross-validation.



Sumber: Verdikha (2018)

Image 4. *Cross Validation* KFold 10

The parameters used in the cross-validation process are:

a. *n_splits* : This parameter determines the number of folds to be created, set to 10.
b. *train_index & test_index* : These are used to obtain the indices of each fold.
c. *Split* () : This method iterates over each fold of the Kfold object, dividing the data into training and testing sets.
d. *Enumerate*() : This function is used to obtain the indeks of each fold.
e. *Train set & Test set* : These are used to store the training and testing data, respectively.

2.3 Feature Extraction FastText

FastText is a word embedding method developed by Facebook AI Research in 2016. It represents words as vectors and extends the Word2Vec model by handling out-of-vocabulary words through sequences of n-grams. Additionally, FastText uses the Bag of Words (BoW) approach for word processing (Nurdin et al., 2020).

FastText features a neural network architecture comprising three components. The input layer consists of word vectors derived from the Bag of Words (BoW) model and n-grams. The hidden layer processes these vectors to compute a set of word embeddings formed from the words. Finally, the output layer averages the vectors from the hidden layer to produce a linear classification result (Herwanto et al., 2019).

The process of forming vector representations for feature extraction using FastText involves the following steps. The tools used include Python with the FastText library (Joulin et al., 2016). The parameters utilized, as explained in the background from the study by Amalia et al. (2020), are as follows:

a. *Learning Rate* : This parameter determines the rate of learning, set to 0.1.
b. *Epoch*: This parameter indicates the number of iterations used to train the data, with a value of 10.
c. *Word n-gram*: This parameter specifies the number of n-grams used in feature formation from the words in the text, set to 2.
d. Model: This parameter is used to store the FastText model.

2.4 Support Vector Machine (SVM) Classification

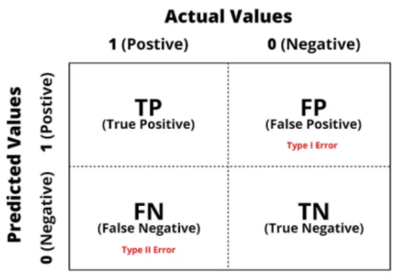
Support Vector Machine (SVM) is a model developed by Vapnik and his team for pattern classification. It employs a technique to manage error boundaries by minimizing the distance between the training data and the hyperplane. This approach contrasts with traditional methods that focus on reducing observational testing errors. The training process for SVM involves solving a quadratic programming problem with linear constraints, which differs from training other networks that involve non-linear optimization and carry the risk of getting trapped in local minima. In SVM, the solution depends only on a subset of the training data known as support vectors (Singh et al., 2020).

To classify the model with SVM, the following parameters are required, using the `sklearn` library (Pedregosa et al., 2011):

a. Kernel: The kernel in SVM transforms the feature space to convert non-linear data into linear data. There are four types of SVM kernels: linear, radial basis function (RBF), polynomial, and sigmoid. In this study, four scenarios will be tested using these different kernels.
b. C: This parameter controls the trade-off on the margin, with a value set to 1.0.
c. Gamma: This parameter controls the shape of the hyperplane, with the value set to "scale”.

2.5 F1 Score

The F1 Score is derived from the confusion matrix, which provides information about the results of classification predictions. It includes four components: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). This is illustrated in Figure 5 below.



Source: Nugroho (2019)

Image 5. Confusion Matrix

a. True Positif (TP): Represent the number of positive cases correctly classified as positive.
b. False Positif (FP): Represents the number of negative cases incorrectly classified as positive.
c. True Negatif (TN): Represents the number of negative cases correctly classified as negative.
d. False Negatif (FN): Represents the number of positive cases incorrectly classified as negative.

The F1 Score combines precision and recall, and can be calculated using the following formula.

$$F1\ Score = 2 * \frac{precision\ x\ recall}{precision + recall} \qquad (1)$$

Where precision is the ratio of true positive predictions to the total number of positive predictions, and recall is the ratio of true positive predictions to the total number of actual positive cases. Therefore, the F1 Score emphasizes the balance between false positives and false negatives, aiming to provide a more balanced measure of classification performance.

This step involves measuring the alignment between the model's classification results and the actual classes using a confusion matrix and the F1 Score, utilizing the sklearn library (Pedregosa et al., 2011). The parameters used for each fold and the average across all folds are as follows:

a. y_test : This represents the actual values of the target or labels in the classification.
b. y_pred: This represents the predicted values for each sample of data from each SVM kernel.
c. np.mean : This is a function used to calculate the average F1 Score across all folds.

3. Results and Discussion

3.1 *Data Cleaning*

Secondary data that has been described in the Research Methods section in the form of Comma Separated Values (CSV) is imported into python. In this data there is 1 Tweet column and 12 class columns. From this dataset there are still built-in attributes from Twitter such as username, retweet, URL and emoji so that it requires a cleaning stage before the experiment is carried out. The columns that will be used in this research are the Tweet column containing text and the HS column containing numbers.

From the HS column there are 2 classes, namely 1 is the HS class and 0 is the Non HS class with a total number of rows of 13169. The number of class 0 columns is 7608 and the number of class 1 is 5561, Figure 6 is a graphical display of the HS column.

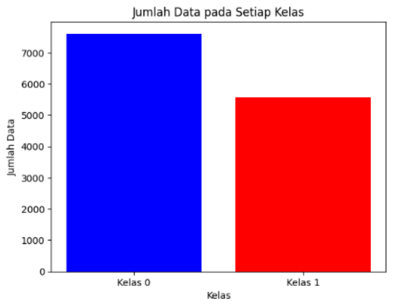


Image 6. Data Count Chart

Then the data enters the preprocessing stage with each computation time result as in Table 1 as follows.

Table 1. Preprocessing Time

Process	Time
Insert Clumn ID	0,005 Seconds
Lower Case	0,008 Seconds
Remove Attribute	0,49 Seconds
Non Alphanumeric	3,98 Seconds
Spell Checker	0,11 Seconds
Stemming	17 Minute
Stopword	10,43 Seconds
Visualisasi Kata	1,96 Seconds
Remove Data Kosong	0,03 Seconds

Based on the results of preprocessing the computation time of each stage is obtained the longest time at the stemming stage, then the results of visualization still have noise words or words that are not included in the HS word. From this stage the initial data with the amount of data 13169 becomes 13112 final data which is then stored in CSV form.

3.2 Data Division

With the data division used, namely Cross Validation, which is carried out before the data enters the formation of the extraction feature model by separating the preprocessing data to be used, namely the Tweet Clean column and the HS column. The following are some of the results of the division of train data and test data from fold 1 and 10 to find out the division of data according to the process.

From the results of fold 1 there is train data with a total of 11800 samples with indices from 1312 to 13111 and test data as many as 1312 samples with indices from 0 to 1311, the time required is less than 0.00 seconds.

While from fold 10 there is train data with a total of 11801 samples with indices from 0 to 13111 and test data as many as 1311 samples with indices from 11801 to 13111, the time required is the same as in fold 1.

3.3 Feature Extraction

The feature extraction process in FastText uses data that has been separated from the cross validation process which is then inputted in the form of a text file that can be read by the FastText model. In the formation of data vectors separated into 2, namely train data and test data in the "x" data column which is the "Tweet_Clean" data column, this is done to ensure that the features generated from the FastText model come from the appropriate data, namely features from training data used to train the model and features from test data used for testing.

Based on the data train features formed for 76.7247 seconds (1.27874 minutes) with a vector of 100 dimensions and the range of vector values obtained is at a value of -0.065 to 0.067, from the value of the vector range can determine the distribution of vector values in the data. Meanwhile, the test data features formed for 10.65 seconds with the same vector dimensions and has a range of vector values obtained at a value of -0.007 to 0.009, from the value of the vector range can determine the distribution of vector values in the data.

3.4 Classification and Evaluation

From this stage the FastText feature enters the classification with the SVM algorithm and calculates the evaluation results to determine the effectiveness of the method from the evaluation results. Where, this process obtains the results of each fold for each SVM kernel, so that the results obtained are as listed in Table 2.

The F1 Score results from each fold and 4 SVM kernels show the highest results are in the Polynomial kernel with a value of 0.826 at fold 8, this indicates that the division of train and test data has better performance than the sigmoid kernel which is 0.792 at fold 4.

When viewed from the average confusion matrix of each SVM kernel, the results will be obtained as shown in Table 3 from each SVM kernel

Table 2. Evaluation of Each Fold

Fold	**F1 Score**			
	RBF	Linear	Polynomial	Sigmoid
Fold 1	0,796	0,800	0,810	0,795
Fold 2	0,809	0,809	0,812	0,802
Fold 3	0,819	0,819	0,809	0,820
Fold 4	0,797	0,798	0,803	0,792
Fold 5	0,804	0,805	0,806	0,803
Fold 6	0,806	0,805	0,825	0,800
Fold 7	0,805	0,808	0,809	0,806
Fold 8	0,817	0,817	0,826	0,816
Fold 9	0,810	0,808	0,800	0,806
Fold 10	0,816	0,819	0,828	0,815

Table 3. Average Confusion Matrix

Kernel	Confusion Matrix			
	TP	FP	TN	FN
RBF	438	122	676	75
Linear	443	117	673	78
Polynomial	484	76	627	124
Sigmoid	440	120	672	79

Based on TP, TN, FP, and FN from each kernel and the calculation of the average F1 Score from each fold using np.mean in the program function, the results are shown in Table 4 as follows.

Table 4. F1 Score Results of Each Kernel

Average F1 Score	
Kernel	Results
RBF	0,808
Linear	0,809
Polynomial	0,813
Sigmoid	0,805

Based on the F1 Score evaluation results for various kernels, the RBF kernel has an F1 Score value of 0.808. This kernel shows good performance in classification, although the F1 Score value is slightly lower than the Polynomial and Linear kernels. Linear kernel has an F1 Score value of 0.809. This kernel provides almost equivalent performance to the Polynomial kernel, showing a good ability to classify data. The Polynomial kernel has the highest F1 Score value of 0.813. It gives the best performance among all the kernels evaluated, indicating that the model with the Polynomial kernel is more suitable for the dataset used in this experiment. The Sigmoid kernel has a slightly lower F1 Score value of 0.805. It performs well, but is slightly less optimal than the Polynomial and Linear kernels.

4. Conclusion

From the analysis, it can be concluded that the combination of FastText and SVM methods on this secondary data obtained good results by considering the use of kernels, where the Polynomial kernel showed the best performance in classifying the data, followed by the Linear kernel. While the RBF and Sigmoid kernels also show good performance, but their F1 Score is slightly lower than the previous two kernels. This indicates that the F1 Score results do not provide a significant difference in results.

Reference

Adhari, A., Nasrun, M., & ... (2021). Deteksi Ujaran Ancaman Berbasis Website Pada Media Sosial Twitter Menggunakan Metode Support Vector Machine. EProceedings …, 8(2), 1920–1925. https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/viewFile/14602/14381

Amalia, A., Sitompul, O. S., Nababan, E. B., & Mantoro, T. (2020). An Efficient Text Classification Using fastText for Bahasa Indonesia Documents Classification. 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics, DATABIA 2020 - Proceedings, 69–75. https://doi.org/10.1109/Databia50434.2020.9190447

Antariksa, K., Purnomo WP, Y. S., & Ernawati, E. (2019). Klasifikasi Ujaran Kebencian pada Cuitan dalam Bahasa Indonesia. Jurnal Buana Informatika, 10(2), 164. https://doi.org/10.24002/jbi.v10i2.2451

Baskoro, F., Andrahsmara, R. A., Darnoto, B. R. P., & Tofan, Y. A. (2021). A Systematic Comparison of Software Requirements Classification. IPTEK The Journal for Technology and Science, 32(3), 184. https://doi.org/10.12962/j20882033.v32i3.13005

Berrar, D. (2018). Cross-validation. Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics, 1–3(January 2018), 542–545. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

Cahyono, S. (2023). Sastrawi: Indonesian Stemmer**.** Retrieved from https://github.com/sastrawi/sastrawi

Chen-Wishart, M. (2014). Python Machine Learning Third Edition. In Vascular (Issue January 2010).

Ibrohim, M. O., & Budi, I. (2019). Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter. 46–57. https://doi.org/10.18653/v1/w19-3506

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). FastText.zip: Compressing text classification models. 1–13. http://arxiv.org/abs/1612.03651

Herwanto, G. B., Maulida Ningtyas, A., Nugraha, K. E., & Nyoman Prayana Trisna, I. (2019). Hate Speech and Abusive Language Classification using fastText. *2019 2nd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2019*, 69–72. https://doi.org/10.1109/ISRITI48646.2019.9034560

Kedia, A., & Rasu, M. (2020). Hands-On - Python Natural Language Processing.

McKinney, W. (2010). Data Structures for Statistical Computing in Python**.** In *Proceedings of the 9th Python in Science Conference* (pp. 51-56). Retrieved from https://pandas.pydata.org/pandas-docs/stable/

Nugroho, K.C., (2019). Confusion Matrix untuk Evaluasi Model pada Supervised Learning. [Online] Tersedia di <https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f> [Diakses 18 Juli 2023]

Nurdin, A., Anggo Seno Aji, B., Bustamin, A., & Abidin, Z. (2020). Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks. Jurnal Tekno Kompak, 14(2), 74. https://doi.org/10.33365/jtk.v14i2.732

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Barupal, D. K., & Fiehn, O. (2011). Scikit-learn: Machine Learning in Python. Environmental Health Perspectives, 127(9), 2825–2830. https://doi.org/10.1289/EHP4713

Python Software Foundation. "re — Regular Expression Operations." *Python Documentation*, version 3.11.4, 2023, https://docs.python.org/3/library/re.html.

Singh, V., Poonia, R. C., Kumar, S., Dass, P., Agarwal, P., Bhatnagar, V., & Raja, L. (2020). Prediction of COVID-19 corona virus pandemic based on time series data using support vector machine. *Journal of Discrete Mathematical Sciences and Cryptography*, *23*(8), 1583–1597. https://doi.org/10.1080/09720529.2020.1784535

Verdikha, N. A., Adji, T. B., & Permanasari, A. E. (2018). Komparasi Metode Oversampling Untuk Klasifikasi Teks Ujaran Kebencian. Seminar Nasional Teknologi Informasi Dan Multimedia 2018, 85–90.