




Improving Machine Learning Performance by Eliminating the Influence of Unclean Data

Murtadha B. Rissan*, Rehab F. Hassan 

Computer science Dept., University of Technology-Iraq, Alsina'a street, 10066 Baghdad, Iraq.

*Corresponding author Email: cs.19.02@grad.uotechnology.edu.iq

HIGHLIGHTS

- From four models to teach machines (SVM, Multiple Bayes - NB, and Bernoulli - NB) used, Best accuracy (Bernoulli - NB) model 89%.
- with the Bernoulli-NB model reaching 91% of accuracy, as well as improving the value of the rest of the models used in this process

ARTICLE INFO

Handling editor: Rana F. Ghani

Keywords: Preprocessing

Data cleaning

Machine Learning

Data mining

ABSTRACT

Regardless of the data source and type (text, digital, photo group, etc.), they are usually unclean data. The term (unclean) means that data contains some bugs and paradoxes that can strongly impact machine learning processes. The nature of the input data of the dataset is the most important reason for the success of the learning algorithm. More than one factor influences machine learning results in a specific task. The characteristics and the nature of the data are the main reasons for the algorithm's success. This paper generally examines data processing entered into an algorithm to learn machines. The paper explains the operations of each stage of prior treatment data for the best achievement of its data set. In this paper, four models for teaching machines (SVM, Multiple Bayes - NB, and Bernoulli - NB) will be used. Best accuracy (Bernoulli - NB) model 89%. The pre-processing algorithm applied to the data set (dirty data) will be developed and compared to previous results before development. The Bernoulli-NB model reaches 91% accuracy and improves the value of the rest of the models used in this process.

1. Introduction

There is a rapid increase in machine learning and deep learning that essentially enable us to build models with predictability. But getting the data right is the most common problem. We have all the sophisticated models, and they're effective, but it will be a real challenge to build a model whose results are accurate because of the unclean data. Workers in this field face the problem of processing unclean data when it comes from more than one source [1]. An enormous amount of digital text information is generated daily, and the effective search for, managing, and exploring text data has become the main task [2]. Usually, data from datasets are unclean. That is, it contains many unwanted characters and symbols, which may negatively affect the quality of learning. Social network data is enormous and usually has multiple sources, making it vulnerable to noise and redundancy. Poor quality data will restrict the analysis and negatively affect the analysis results. Therefore, it is imperative to clean up the data sets before starting the machine learning process. [3]. This research paper aims to provide a semi-detailed overview of the different stages, focusing on the data obtained from social media sites. Pre-processing (handling unclean data) is considered an effective technique to ensure the validity of the collected data. So pre-processing unclean data removes all punctuation marks, words that have no meaning, and words can be grouped into groups, and words or lemma can be truncated to their roots, depending on the pre-processing purpose applied to the dataset [4,5,6].

2. Related work

In 2017 Priyanga Chandrasekar and Prabir Bhattacharya presented improving the Prediction Accuracy of Decision Tree Mining with Data Preprocessing. They used J48 for classification and Weka as a data mining tool. They utilized the supervised filter discretization on the J48 algorithm for constructing the decision tree. Then, they compared the results with the J48 without discretization. The results from experiments showed that the accuracy of J48 after discretization is better than J48 before discretization [7]. In 2017 Nerijus Paulauskas and Juozas Auskalnis. Present Analysis of Data Pre-processing Influence on Intrusion Detection using NSL-KDD Dataset. The work analyses the initial data pre-processing effect to attack detection accuracy by Naïve Bayes, using Decision Trees and Rule-Based classifiers with the NSL-KDD dataset. The data set for method

analyses were selected NSL-KDD. The data set consists of 125973 network flow data, 41 features, and one class marked as normal or attack. There are 24 different attack types in the train data set. Used for ensemble model of four different classifiers: J48, Naïve Bayes, C5.0, and PART with and without pre-processing, using only the train data set and dividing it into 70% for training and 30% for testing [8]. In 2018 Ammar Ismael Kadhim. Present an Evaluation of Preprocessing Techniques for Text Classification. Two different methods, TF-IDF and chi-square with a cosine similarity score, are used to extract the features based on the BBC English dataset, an English dataset collected manually from the BBC online newspaper. The dataset consisted of 4,470 articles published online from 2012 to 2013. TF-IDF's performance with the cosine similarity score was better in classifying the ten general categories based on evaluation metrics. The results showed that pre-processing of the text can improve text recognition and the performance of the text classification system. In 2019 M. Sornam and M. Meharunnisa. Present The discovery of normality of body weight using principal component analysis (A comparative study on machine learning techniques using different data pre-processing methods). This research aims to perform feature selection by utilizing the PCA to determine women's normal and abnormal body weight. The major components obtained are passed as input to the supervised learning algorithm, such as support vector machine, K-nearest neighbor, decision tree, naïve Bayes, and backpropagation neural network with several pre-processing methods. The dataset is gained using the survey cum questionnaire method through the expert and document review through 37 questions. The imputed data are pre-processed using several methods. The feature reduction on normalized data is then made by PCA. Thirty-seven features are reduced to 29. The highest obtained accuracy is for BPN, SVM, KNN, SVM, and KNN, respectively, with 73%, 74%, 70%, 76%, and 77% [9].

3. Main steps of Pre-processing stages

The following are the main steps in any pre-processing stages, as shown in Figure 1 and will be discussed in the following subsections:

- 1) Extraction (Dataset construct)
- 2) Text Cleaning
- 3) Tokenization
- 4) remove Stop-Words
- 5) Stemming
- 6) Parts of Speech (POS) Tagging
- 7) Feature Extraction

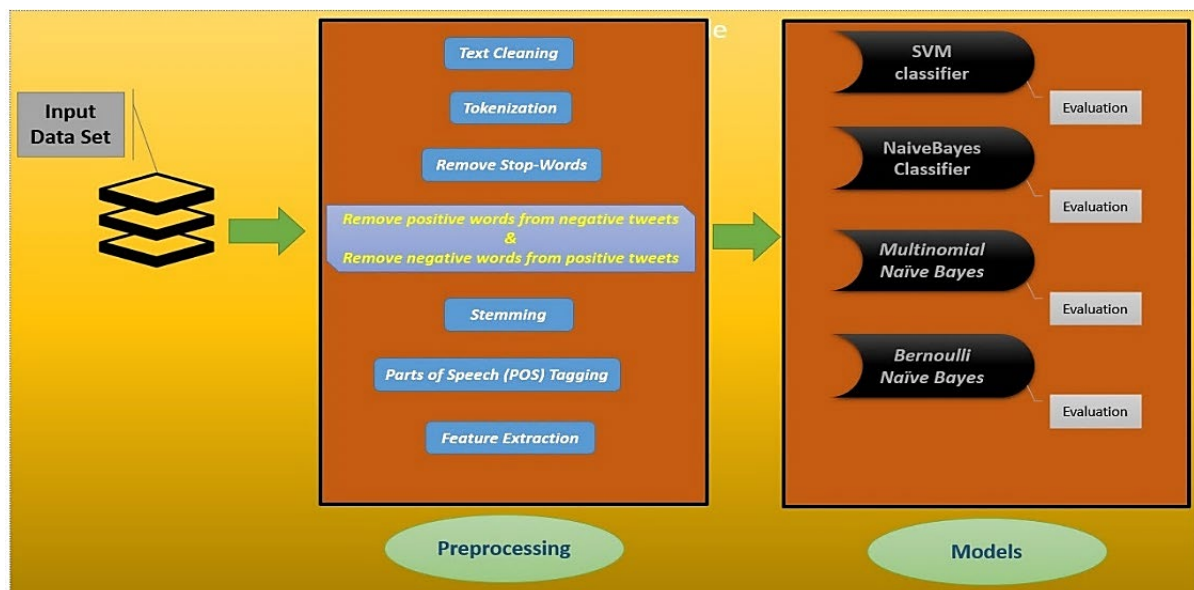


Figure 1: Main Steps Of Pre-processing Stages

3.1 Extraction (Dataset construct)

The data set is either built or extracted from an application designed for this purpose or taken readily from a specialized website. For example, a dataset of Twitter tweets is either developing an application to get tweets according to specific hashtags through the API window or taking tweets in a ready-made data set from one of the sites that publish them [10]. Two files containing tweets (positive tweets and negative tweets) were used as a dataset in this research. These samples were collected from Tweets (or "status updates") from Twitter Streaming APIs. Each file consists of JSON tweets, separated by a line, i.e., one tweet per line. These tweets were collected during July 2015. The total number of positive tweets in the file ("positive_tweets.json") is 5,000, and the same in the file ("Negative_tweets.json"). She also has 5,000 tweets. The total size of the data set is 10,000 Tweets. Any data set can also be fetched, and the same working steps apply below. You can get Tweets via the Twitter API or from any other source for the dataset.

3.2 Text cleaning

punctuation or special symbols that occur in sentences such as ("([0-9+])(#)((@[A-Za-z0-9+])([^\0-9A-Za-z\t])((w+:\V\S+))") or some English abbreviations such as (n't = not or I'm = I am or OMG = Oh My God) Taking this move-in facilitation is quite beneficial and practical. The main goal of Usually, unnecessary symbols, tokens, and noise that must be deleted before any other normalization technique operations are included in the text data taken from the electronics websites used for the study. An example of extracting text from it is data sources, such as HTML data containing not needed HTML tags or any data from XML and JSON files [11]. If the NLP (Natural Language Processing) application programmer does not handle the text appropriately, the results will be unwanted or irrelevant. Text processing is cleaning the text, which supports the rising accuracy of classifiers [12]. The elimination of unnecessary and special characters is one of the most important functions during text normalization. This may be this step is that Natural Language Processing, and machine learning results are affected by punctuation or special characters when performing text analysis and extracting features or information [11].

3.3 Tokenization

Text data generally is a group of letters in the primary stage. each operation in the text analysis stage will need the words of the dataset. The requirements of the parser are tokenization documents [12]. Tokenization separates raw text data into several individual tokens, which are represented as a word or character [13]. The purpose of tokenization is to research phrases in a sentence (or phrase). An input for other operations such as analysis or text mining would be the token list. In both linguistics (where it is a type of text segmentation) and computer sciences, which is part of lexical analysis, tokenization is useful. Text data is a piece of characters (or letters) in the first stage. It requires processing and converting all words into lowercase. Tokenization aims to identify meaningful keywords. Another problem is the abbreviations and acronyms that must be converted to a standard format [10]. There are two types of tokenization:

3.3.1 Sentence tokenization

Sentence tokenization is a way to break a textual document into a group of sentences. The goal of this operation is to make the texts into meaningful sentences. The techniques used to search for separate between sentences to achieve sentence tokenization, like a duration (.) or a newline character (\n) [11,12].

3.3.2 Word tokenization

Work tokenization divides the sentence into words that make up that sentence, called tokens. These tokens can be used in cleaning and normalization, such as derivation and conjugation [12]. There are several resources available for the tokenization of textual documents. The resources are:

- 1) Mila Tokenizer
- 2) Nlpdotnet Tokenizer
- 3) NLTK Word Tokenize
- 4) TextBlob Word Tokenize
- 5) Pattern Word Tokenize
- 6) MBSP Word Tokenize
- 7) Tokenization of Words with Python (NLTK)

(Tokenization of Words with Python NLTK) used to work for this paper. NLTK stands for Tool Kit of Natural Language. It's the Toolkit for Python Natural Language Handling. NLTK is a good platform for developing Python programs to deal with human languages data. In addition to a group of libraries for word processing for tokenization, grouping, tagging, semantic reasoning, and stemming, NLTK offers many easy-to-use tools for more than 50 classes (the body collection is a broad and structured collection of texts) and lexical resources such as WordNet [13].

3.4 Remove stop-words

Stop words, stop words as some references mention them. Stop words are parts of natural language. Words that have little matter or no meaning. A stop word group of usually frequented features appears in every text document. Common features like conjugations such as and, or, but, pronouns, she, she, etc., should be removed because they have little if no effect on the text mining process. Due to the high frequency of their occurrence, the appearance of these words makes it difficult to understand the content of the text files that contain them. Removing stop words from a document text is to sort the text to its appearance more order by eliminating the less important word. This process decreases the amount of processed textual data, which gets better system performance [3, 14, 15].

3.5 Stemming

The method of removing suffixes from a word to obtain the base of the word. Affixes are units like prefixes, suffixes, etc., which are attached to a word's stem to change the word's meaning or create a completely new word. Stemming is used to reduce the number of features in the feature space, improve classifier performance, and assist with the machine learning process when variants of features are rooted in a single feature. For example: (connect, connects, connected, and connecting). Stemming helps standardize words to their basic origins regardless of their conjugations, enabling many applications such as text classification, grouping, and machine learning operations. The nltk package contains many applications for derivatives. One of the most popular

derivatives is Porter Stem, which is based on the algorithm built by Dr. Martin Porter. The Porter trunk runs in the following code snippet: (from nltk.stem import PorterStemmer) [9, 11].

3.6 Parts of Speech (POS) tagging

Part of speech (POS) tagging: is a way in which a part of speech specifies every word in a phrase. POS knowledge plays a fundamental role in sentence checking because every word in a sentence with each POS tag has a different meaning depending on the sentence situation. Parts of speech (POS) are unique lexical types the words during POS operation are assigned depending on their syntactic meaning and purpose. The primary speech components include verbs, nouns, adverbs, and adjectives [1]. Table 1 shows some examples of parts of speech. POS tags can differentiate words and articulate their parts of speech. It is very helpful, especially when NLP-based applications require using the same annotated text. It can be filtered by identifying parts of speech and using this knowledge to continue with precise research, such as narrowing names and figuring out which ones are most relevant [11].

Table 1: Examples of Tags with Parts of Speech

TAG	Descriptions	Example(s)
NN	The singular or collective noun	lion, cat
VB	Base form - Verb	jump, run
IN	Secondary conjugation - Preposition	That, in, of, on
DT	Determiners	an, a, the
FW	Strange Words	Pon, d'hoove
JJ	Adjectives	rich, smart
JJR	Comparative - Adjective	better, stronger
JJS	Adjective - superlative	fastest, purest
CD	Amount Cardinal	seven, four, 9
NN	A singular or collective noun	lion, cat

3.7 Feature extraction

Feature extraction plays an important role in minimizing errors in machine learning performance. Increases data diversity by creating new features by transforming data to reduce dimensions or adding domain knowledge to improve the algorithm's work [16]. Document pre-processing (dataset) produces a document that contains only a bag of words. To which the algorithms cannot be directly applied. This bag of words should be converted into the term vector. It gives a vector term with numerical values corresponding to each term appearing in a document and is very helpful in feature selection [17]. Two techniques for feature extraction will be explained in this paper.

3.7.1 Bag of Words model

The simplest and oldest form of text feature extraction in NLP is a bag of words. This model converts every document or text into a vector with the number of occurrences of every word in the document. The problem with this model is that it may give great importance to a word that frequently appears in all documents, and ignoring the other may be unique to distinguish such a document. Other models have been suggested to deal with this problem [12].

3.7.2 Tf-idf

Term Frequency Inverse Documents Frequency (TF-IDF) was first-proposed by Karen [18]. The (TF-IDF) model is the most useful and popular one for converting terms into a vector: Term Frequency, Term Occurrences, and Term Frequency inverted document frequency (TF-IDF) [19]. The substantial idea of TF-IDF is that terms that appear frequently in many documents are considered less important than high-frequency words that appear within one document. TF-IDF is calculated by Eq (1).

$$TF - IDF = \frac{nt}{N} \cdot \log \frac{K}{Kn} \quad (1)$$

In this equation, nt is the occurrence of term t within a document, N is the number of terms in the document, K is the total number of documents, and Kn is the number of documents that contain the term t [19].

4. Machines learning

The work of this paper applied to the mentioned data set (positive and negative tweets) using the SVM (Support Vector Machine) classifier and family naïve bays classifiers. The popular Naïve Bayes Classifiers Are Multinomial Naïve Bayes Classifier and Bernoulli Naïve Bayes Classifier.

4.1 Support vector machines

SVMs are classifiers that label and classify data in the feature space [20]. Support Vector Machines (SVM) are supervised learning algorithms applied for regression, classification, and detection of anomaly outliers. In case of data had been trained when each data point in the data set belongs to a particular class, according to a binary classification problem, the Support Vector Machines algorithm can be trained based on this data so that each data point can be assigned to one of these two classes [9].

4.2 Naïve Bayes

A Naïve Bayes classifier is a probabilistic classifier depending on the Bayes theorem with an independence assumption. It is supervised learning and will be trained very efficiently [21]. It can learn Class membership probabilities, like expecting the probability of a given value belonging to a particular class. The algorithm is a basic Bayesian classifier name the Bayesian naive classifier. If utilized in large datasets, Bayesian classifiers results showed high speed and accuracy. Naive Bayesian classifiers presume that the influence on a given class of an attribute value is distinct from the other attribute values. This case is named class conditional independence. It is constructed to simplify the computations [22].

4.3 Multinomial naïve bayes

Multinomial NB is similar to Naïve Bayes and is also a probabilistic tactic. Multinomial NB develops the utilization of the Naïve Bayes algorithm. It uses NB for data partitioned multinomially and is a frequency-depended model. The Multinomial Naïve Bayes algorithm operates on the definition of the frequency of the term, explaining how many times repeated the item is during operation. The main variation between classifiers of Multinomial Naïve Bayes and Naïve Bayes Generally, Naïve Bayes (NB) works based on conditional probability (the conditional independence of the characteristics is considered). In contrast, the Multinomial Naïve Bayes works based on the multinomial distribution. Multinomial Naïve Bayes also can be said as an upgraded version of the Naïve Bayes algorithm. It effectively calculates an item's frequency [23-25].

4.4 Bernoulli naïve bayes

Bernoulli's Naïve Bayes algorithm works well on the binary concept that when the items are repeated or not, but not similar to Multinomial Naïve Bayes, it does not notify the term frequency. Bernoulli Naïve Bayes, unlike the multinomial process in that the term frequencies are taken into account by the multinomial approach. In contrast, the Bernoulli approach is only interested in determining whether or not a term is present in the text under consideration. In the multivariate Bernoulli Naïve Bayes algorithm, features are distinct binary variables, which explain if the term appears in the file under specified consideration or does not [23,26,27].

5. Results and Discussion

After dividing the dataset into Training and Testing at a ratio of (70:30) and training the machine on this dataset, two algorithms summarize pre-processing steps below. Algorithm No. 2 was proposed as an algorithm to work because its results are better than algorithm No. 1, as we can see through the results tables. As we can see through the results tables, where Table 2 shows the results of the work of algorithm 1, and Table 3 concerns the results of algorithm 2. A total of 2,781 negative and 2005 positive words were collected in the English language. Positive words were placed in the file extension txt, and negative words were placed in FileText. In the program, a special step breaks up one tweet into several words and compares these words with the words in the file, meaning that the positive tweet words are compared with the negative words in the file. If it finds a negative word within the positive tweet, the program will delete this word, as is the case with tweets. The negative word is where the negative words of the tweet are broken down and compared with the words in the file for the positive words. If he finds a positive word within a negative tweet, he will delete this word from the tweet. As mentioned in the attached tables, this step increased accuracy and improved results. The results of the updated algorithm (Algorithm 2) are as follows.

Table 2: Results of Algorithm1

Metric/algorithm	SVM	Naive-Bayes	Multinomial - NB	Bernoulli - NB
Accuracy	86	83	82	89
Precision (positive tweets)	81	82	82	82
Precision (negative tweets)	78	83	80	80
Recall (positive tweets)	89	84	84	84
Recall (negative tweets)	93	82	87	87
F1-Score (positive tweets)	85	83	83	83
F1-Score (negative tweets)	85	83	84	84

Table 3: Results Of Algorithm 2

Metric/algorithm	SVM	Naive-Bayes	Multinomial - NB	Bernoulli - NB
Accuracy	88	86	85	91
Precision (positive tweets)	83	93	84	84
Precision (negative tweets)	79	80	80	80
Recall (positive tweets)	94	77	89	89
Recall (negative tweets)	95	94	94	94
F1-Score (positive tweets)	88	84	87	87
F1-Score (negative tweets)	87	87	86	86

Figure 2 shows the comparison between Algorithm No. 1 and Algorithm No. 2 (results of each machine learning).

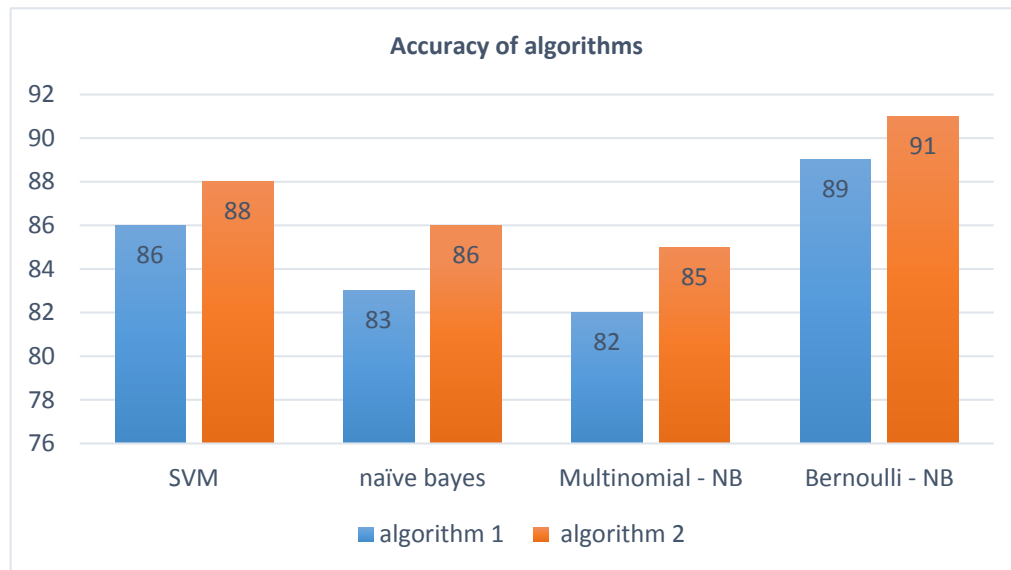


Figure 2: Accuracy levels after the algorithm update

6. Conclusion

In this paper, more than one step is used (cleaning, removing stop words, tokenization, Stemming, part of speech tagging, and feature selection) to apply it to the dataset to make the dataset clearer and less error-free, and free from noises as much as possible. Using other machines and training it or a different data set is possible to get better accuracy and results. After applying these steps, it was observed that the accuracy of machine learning increased when removing negative words from positive tweets and removing positive words from negative tweets, where the accuracy (Bernoulli - Naive Bayes) became 91 after what was 89. The accuracy of (SVM) increased from 86 to 88. The accuracy of the results depends on the data set used and the model used in machine learning.

Algorithm 1:

Input: source dataset (uncleaned and unstructured tweets)

Output: cleaned and labeled tweets

Start

Step1: Read the data set (5,000 positive tweets and 5,000 negative tweets)

Step2: Fragmentation \ tokenize the tweets, each tweet separately, then tokenize the tweet into words.

Step3: Remove noise \ Clean each tweet (token) from numbers, punctuations, and handling abbreviations.

Step4: Remove stop words.

Step5: POS \ Part of Speech tagging each remained word.

Step6: Stemming: return the verb to it is root.

Step7: Mark [Label] each tweet as positive or negative

End

Algorithm 2

Input: source dataset (uncleaned and unstructured tweets)

Output: cleaned and labeled tweets

Start

Step1: Read the data set (5,000 positive tweets and 5,000 negative tweets)

Step2: Fragmentation \ tokenize the tweets, each tweet separately, then tokenizes the tweet into words.

Step3: Remove noise \ Clean each tweet (token) from numbers, punctuations, and handling abbreviations.

Algorithm 2: Continued**Algorithm 2****Input:** source dataset (uncleaned and unstructured tweets)**Output:** cleaned and labeled tweets**Step4:** Remove stop words.**Step5:** Remove Negative words from positive tweets & Remove positive words from negative tweets (by comparison, each word in a tweet with external files contains positive and negative words).**Step6:** POS \ Part of Speech tagging each remained word.**Step7:** Stemming: return the verb to it is root.**Step8:** Mark [Label] each tweet as positive or negative**End****Author contribution**

All authors contributed equally to this work.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author.

Conflicts of interest

The authors declare that there is no conflict of interest.

References

- [1] S. P. Kumar, S. Bakshi, I. K. Hatzilygeroudis, M. N. Sahoo, Recent Findings in Intelligent Computing Techniques, Proc., 3 (2017). <https://doi.org/10.1007/978-981-10-8633-5>
- [2] H. Attya, Y. H. Ali, A. Abdulwahab, Documents Classification Based On Deep Learning, Int. J. Sci. Eng. Res. Biosci. Bio.technol. Res., 9 (2020)62-66.
- [3] A.Q. Albayati, A. S. Al-Araji, S. H. Ameen. A. Method of Deep Learning Tackles Sentiment Analysis Problem in Arabic Texts, Iraqi J. Comput. Control. Syst. Eng., 20 (2020) 9-20.
- [4] V. Kalra, R. Aggarwal, Importance of Text Data Preprocessing & Implementation in RapidMiner, Ann. Comput. Sci. Inf. Syst.,14 (2018) 71-75. <http://dx.doi.org/10.15439/2017KM46>
- [5] A. I. Kadhim, An Evaluation of Preprocessing Techniques for Text Classification, Int. j. comput. sci. inf. Secur., 16 (2018) 22-32.
- [6] D. H. Abd, A.T Sadiq, A. R. Abbas, Classifying political arabic articles using support vector machine with different feature extraction, Appl.Comput. Support Ind. Innov. Technol., 1174 (2019) 79–94. https://doi.org/10.1007/978-3-030-38752-5_7
- [7] P. Chandrasekar, K. Qian, H. Shahriar, P. Bhattacharya, Improving the prediction accuracy of decision tree mining with data pre-processing, IEEE Annual Comp. Soft. Appl. Conf., 2 (2017) 481-484. <https://doi.org/10.1109/COMPSAC.2017.146>
- [8] N. Paulauskas, J. Auskalnis, Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset, Open Conf. Electr. Electr. Info. Sci., (2017) 1-5. <https://doi.org/10.1109/eStream.2017.7950325>
- [9] M. Sornam, M. Meharunnisa, The discovery of normality of body weight using principal component analysis: a comparative study on machine learning techniques using different data pre-processing methods, Int. J. Knowl. Eng. Data Mining, 6 (2019) 74-88. <https://doi.org/10.1504/IJKEDM.2019.097356>
- [10] S. Vijayarani, J. Ilamathi, M. Nithya, Pre-processing techniques for text mining-an overview, Int. J. Comput. Sci. Commun. Netw., 5 (2015) 7-16.
- [11] S. Dipanjan, Text Analytics with Python, A Practical Real-World Approach to Gaining Actionable Insights from your, Data 2016. <https://doi.org/10.1007/978-1-4842-2388-8>
- [12] A. B. Abdul-Wahhab, A. K. Abdul- Hassan, Opinion Extraction Framework Using Aspect Based Sentiment, Phd. Diss. Comput. Sci. Dept., 2019.

- [13] S. Vijayarani, R. Janani, Text mining: open source tokenization tools-an analysis, *Adv. comput. intell. int. j.*, 3 (2016) 37-47. <https://doi.org/10.5121/acii.2016.3104>
- [14] S. Qaiser, R. Ali, Text mining: use of TF-IDF to examine the relevance of words to documents, *Int. J. Comput. Appl.*, 181 (2018) 25-29. <https://doi.org/10.5120/ijca2018917395>
- [15] R. M. Hadi, S. H. Hashem, A. T. Maolood, An Effective Preprocessing Step Algorithm in Text Mining Application, *Eng. Technol. J.*, 35 (2017) 126-131. <https://doi.org/10.30684/etj.2017.138624>
- [16] H. R. Arabnia, Kevin Daimi. *Principles of Data Sci.*, 2020.
- [17] V. Kalra, R. Aggarwal, Importance of Text Data Preprocessing & Implementation in RapidMiner, *Proc. Int. Conf. Inf. Technol. Knowl. Manag.*, 14 9 (2017) 71-75. <http://dx.doi.org/10.15439/2017KM46>
- [18] S. Robertson, Understanding inverse document frequency: on theoretical arguments for IDF, *J. Doc.*, 60 (2004) 503-520. <https://doi.org/10.1108/00220410410560582>
- [19] F. Zhang, H. Fleyeh, X. Wang, M. Lu, Construction site accident analysis using text mining and natural language processing techniques, *Autom. Constr.*, 99 (2019) 238-248. <https://doi.org/10.1016/j.autcon.2018.12.016>
- [20] Z. K. Alkhateeb, A. T. Maolood, Machine Learning-Based Detection of Credit Card Fraud: A Comparative Study, *Am. J. Eng. Appl. Sci.*, 12 (2019) 535-542. <http://dx.doi.org/10.3844/ajeassp.2019.535.542>
- [21] S. H. Hashim, Proposed Hybrid Classifier to Improve Network Intrusion Detection System using Data Mining Techniques, *Eng. Technol. J.*, 38 (2020) 6-14. <http://dx.doi.org/10.30684/etj.v38i1B.149>
- [22] M. Z. Al-Taie, S. Kadry, J. P. Lucas, Online data pre-processing: a case study approach, *Int. J. Electr. Comput. Eng.*, 9 (2019) 2620-2626. <http://dx.doi.org/10.11591/ijece.v9i4.pp2620-2626>
- [23] G. Singh, B. Kumar, L. Gaur, A. Tyagi, Comparison between multinomial and Bernoulli naïve Bayes for text classification, *Int. Conf. Autom. Comput. Technol. Mngmt.*, (2019) 593-596. <https://doi.org/10.1109/ICACTM.2019.8776800>
- [24] G. P. Wiratama, A. Rusli. Sentiment Analysis of Application User Feedback in Bahasa Indonesia Using Multinomial Naive Bayes, *Int. Conf. New Media Studies.*, (2019) 223-227. <https://doi.org/10.1109/CONMEDIA46929.2019.8981850>
- [25] P. P. M. Surya, L. V. Seetha, B. Subbulakshmi, Analysis of user emotions and opinion using Multinomial Naive Bayes Classifier, *Int. conf. Electr. Comm. Aerospace Technol.*, (2019) 410-415. <https://doi.org/10.1109/ICECA.2019.8822096>
- [26] S. S. Bafjaish, Comparative Analysis of Naive Bayesian Techniques in Health-Related For Classification Task, *J. Soft Comput. Data Mining*, 1 (2020) 1-10.
- [27] M. I. Khalaf, Applied Computing to Support Industry: Innovation, Technol. First Int. Conf., 2019 (2020).