Engineering and Technology Journal
Journal homepage: https://etj.uotechnology.edu.iq

Textual Dataset Classification Using Supervised Machine Learning Techniques

Hanan Q. Jaleel [a], **Jane J. Stephan** [b*], **Sinan A. Naji**

[a] Baghdad College of Medical Sciences, Baghdad, Iraq.
[b] University of Information Technology and Communications, Baghdad, Iraq.
*Corresponding author Email: mailto:janejaleel@uoitc.edu.iq

HIGHLIGHTS

- Automatic text classification has been a significant research domain because of the increased volume of text datasets and documents.
- Any text-based problem should be converted into a form that can be modeled.
- The input text is converted into features using Feature Extraction - Inverse Document Frequency TF-IDF technique.
- Then, five supervised classification methods are used to classify the product's textual keywords into individual classes.
- The suggested technique shows that the Random Forest algorithm is the best and ideal classifier utilized to categorize the dataset with the highest accuracy.

ARTICLE INFO

Handling editor: Rana F. Ghani

Keywords: Bernoulli Naive Bayes; Decision Tree; KNN; Multinomial Naive Bayes; Naive Bayes; Random Forest; Supervised Machine Learning; Text Classification; TF- IDF

ABSTRACT

Text classification has been a significant domain of study and research because of the increased volume of text datasets and documents available in digital format. Text classification is one of the major approaches used to arrange digital information via automatically allocating text dataset records or documents into predetermined classes depending on their contents. This paper proposes a technique that implements supervised machine learning algorithms such as KNN, Decision tree, Random Forest, Bernoulli Naive Bayes, and Multinomial Naive Bayes classifiers to classify a dataset into distinct classes. The proposed technique combines the above-mentioned machine learning classifiers with the TF-IDF feature extraction method as a vector space model to achieve more precise classification results. The proposed technique yields high accuracy, precision, recall, and f1-measure metric values for all the implemented classifiers. After comparing the obtained results of different classifiers, it is found that the Random Forest classifier is the best algorithm used to classify the textual dataset records with the highest accuracy value of 0.9995930.

1. Introduction

The Text Classification (TC) technique acquires more significance and an important role due to the many accessible text documents available from various web sources [1, 2]. Automated text classification into predetermined classes is considered a pivotal technique to handle a massive volume of text documents that are popular, widely spread, and frequently incremented. Digital information can be text documents, text queries, dataset records, publications, web pages, emails, etc. The major challenge is that this huge amount of information requires an arrangement to make it easy to process and manage [1, 2, 3]. Text classification is considered one of the main approaches utilized for organizing digital information. Text classification is a significant subfield of the text mining domain; TC is employed to establish automatic classification techniques using knowledge engineering methods [4, 5, 6]. For example, to automatically label every incoming document with a subject like "art", "politics" or "sport", a data mining classification technique starts with a training dataset D=($d_1,d_2,\ldots\ldots,d_n$) of records or documents that are previously labeled with classes C_1, C_2,…, etc. The classifier model assigns the correct class to the new test document or dataset records d_i. Text classification can be categorized into a single label and multi-label classification [6]. A single label text document can belong to one class only, while a multi-label document can be a member of more than one class [6, 7].

http://doi.org/10.30684/etj.v40i4.1970
Received 9 January 2021; Accepted 23 March 2021; Available online 25 April 2022
2412-0758/University of Technology-Iraq, Baghdad, Iraq

A Typical classification process can be expressed as follows: using a dataset of indexed documents or text records, split it into two parts, a training and a testing set. The classifier attempts to classify the new testing document, depending on its properties, the classification result is the predicted class from the training dataset, that is most similar to the tested example [8, 9, 10].In this paper, the proposed technique implements five supervised classification methods, which are K-Nearest Neighbor KNN, Decision Tree, Random Forest, Bernoulli Naive Bayes, and Multinomial Naïve Bayes classifiers, to classify a product keywords dataset with 65,500 records to evaluate the performance metrics of classifiers and compare the results. The proposed technique determines the best classifier with the highest accuracy, precision, recall, and f1-measure values. The paper's organization is as follows: section 2 introduces the related work of applying the classifiers in classifying datasets and example test data. Section 3 illustrates the proposed technique design and implementation. Section 4 explains and discusses the proposed technique results, and finally, section 5 provides a conclusion to the paper.

2. Related works

Some many researchers and authors work and perform text classification techniques using supervised machine learning algorithms. Manjotho et al. [10] proposed a mobile SMS classifier model that utilizes TF-IDF, and multinomial naive Bayes have been presented. The proposed method aimed to enhance the weighting scheme TF-IDF accuracy and resolve the Naive Bayesian Spam Email classifier's efficiency in classifying mobile messages (SMS) into various classes such as urgent greetings, and invitations, etc. In the training phase, the classifier was learned with 5574 messages. Various preprocessing steps were performed. In the testing phase, the classifier examines 852 messages of the testing dataset. The multinomial naive Bayes classifier was utilized to classify the new test message to the appropriate class. The Multinomial Naïve Bayes classifier provides good accurate results with a TP rate of 93.77 %, an overall accuracy of 93.74%, and a precision of 94.03%. Abbas et al. [11] proposed a technique to classify text movie reviews and implemented sentiment analysis using multinomial naive Bayes. The proposed framework combined the Multinomial Naïve Bayes classifier technique with the TF-IDF approach. They have utilized a movie reviews dataset, where each review involves a textual notice and numerical value (from 0 to l00 scales). They have applied several experiments with many broadly used datasets to evaluate the proposed technique efficiency and yielded good accuracy values. Mowafy et al. [4] introduced a model that uses the logical series of document classification steps and applies the multinomial naive Bayes combined with the TF-IDF approach for text document classification. The experimental results of the 20-Newsgroups dataset have been confirmed via statistical metrics such as recall, precision, and f-score. The results verify that the proposed technique can significantly enhance classification performance. Noor man shah et al. [12] provided a document categorization using the decision tree technique that outcomes in a high accuracy value of classifying text documents to their classes. The proposed approach combined the TF- IDF with a decision tree, where TF- IDF is utilized to assort all words from most repeated to less repeated words. In contrast, the decision tree algorithm is used to make decisions to assign the document to the correct class.

In Singh et al. [13], a classification technique of inter and intra news has been provided and multiple feature-based news classifiers have been suggested depending on the BBC news dataset. For intra-class classification, sports class has been chosen, while the classes of sports, technology, politics, entertainment, and business have been selected for inter-class classification. After applying preprocessing steps, the feature extraction TF- IDF method was implemented. The TF-IDF values of unigram, bigram, and trigram words were utilized as linked or chained feature vectors. The decision tree classifier provides more efficient results with a high accuracy rate. In [14], the researchers presented a framework that uses the KNN method with the TF-IDF extraction approach for text classification. The proposed framework performs text classification using different measurements, parameters, and results analysis. The framework evaluation emphasized the quality and speed of classification. Experimental testing results illustrated the good and bad characteristics of the framework. Jabbar et al. [17] proposed a new technique that integrates the KNN algorithm with a Genetic Algorithm GA for efficient classification of heart disease. GA achieves comprehensive searching in sophisticated massive and multimodal landscapes and supplies optimum solutions.

The experiment results depict that the proposed method improves the accuracy results in diagnosing heart disease and is performed on 7 machine learning data sets. Guo et al. [18] proposed a technique that combines the strong points of the KNN and Rocchio algorithm and implements a proposed model that integrates Rocchio, KNN, and support vector Machine SVM. The proposed technique is performed on two popular document corpuses: the 20-news group and Reuters-21578 news story collections. The experiment results prove that the suggested model surpasses the Rocchio and KMN and is analogous to SVM, which is utilized as a measurement in the experiment. Sjarif et al. [19] present a detection technique that combines TF-IDF and random forest on SMS messages. The proposed technique outperforms a good accuracy, precision, and f1-measure compared to other methods. Various techniques support different outcomes depending on the utilized features.

3. Classification algorithms

3.1 K-nearest neighbor (KNN) classifier

The first step in implementing the KNN classifier is to load the training set feature vectors that represent the weights of keywords. Then, the training phase of KNN saves the training set feature vectors with their related class labels. In the testing phase, the algorithm determines the *K* value, which can be estimated by trial and error and find out the optimal *K* value like 2,3 or 5, or it can be identified based on the dataset by using the following equation [14, 15, 16]:

$$K = \mathrm{sqrt}(N) \quad (1)$$

where N represents the number of data points in the training dataset. For the feature vectors in the testing set, the algorithm gets every test feature vector. It computes the distance between the testing feature vector and all trained data feature vector weights using the Euclidian distance as follows [14, 17, 18]:

$$\text{dist(p,q)= dist (q,p)} = \sqrt{(x-a)^2+(y-b)^2} \quad (2)$$

The Euclidian distance is utilized as a distance measurement because it is the most popular approach. Then, the calculated distances are sorted in ascending order. The top K tuples are obtained from the sorted list, representing the *K* nearest neighbors to the tested feature vector. The algorithm determines the tested feature vector predicted class using the majority vote of the nearest neighbor's class, i.e., get the most frequent class from the nearest neighbors. The most repeated classes will be assigned to the testing feature vector record [14,15,17].

3.2 Decision tree classifier

The Decision Tree constructs the classifier model in a tree structure form. It splits the dataset into smaller subsets. Meanwhile, the related Decision Tree is incrementally improved and progressed. The Decision Tree classifies training textual records by arranging them to be predicted on feature values. Every node in the Decision Tree expresses a feature in the training record to be relegated, and every branch exemplifies a node value. The Decision Tree can be established using a top-down method that recursively divides the training data on the feature that better relegates the training text records. The Decision Tree concept starts by subdividing the data depending on the feature value that is most useful in data relegation. The feature that better splits the trained data will represent the tree root. The decision tree algorithm is iterated on every divided data partition, producing subtrees till the training data is subdivided into subsets of similar classes. At each stage of the partitioning task, a statistical criterion known as Information Gain (IG) is used to decide what features best split the training text records [3, 12]. The DT classifier implies two phases:

1) The training phase, which involves building the decision tree, as the major objective is discovered at each one of the tree's internal or decision nodes, is the ideal testing feature which decreases the classes mixing with each one of the subsets that are generated by the test.
2) The testing phase, which starts with the root of the tree, after that, the feature that is determined by the root node is tested. The test outcome allows moving downward the tree branch related to the given tested record feature. This process is repeated until the algorithm meets a leaf node. This is the reason that the tested record sample is classified by tracking down a route from the root to the leaf nodes. The Decision Trees improve the greedy search algorithm that applies a heuristic function using probability values for comparison [6, 13].

3.2.1 Information Gain and Entropy Measurer

A measurement utilized from the information theory field in constructing a decision tree is Entropy. It has been proven that the Entropy is close and relative to information, which means that the higher the entropy value, the more needed information to characterize that data totally. In constructing Decision Trees, the objective is to decrease the dataset entropy till it reaches leaf nodes, such that the subset holds zero Entropy and represents all examples of one class, i.e., all data examples involve the same value of the target feature or class value. To compute the Entropy of the data set S, the following formula can be utilized [12]:

$$Entropy(Set) = -\sum_{i=1}^{k} P(value_i).log_2(P(value_i)) \quad (3)$$

Pi is the ratio of examples (dataset records) in the dataset that takes the i_{th} value of the class (or target feature), which contains various values. To calculate the information gain (entropy reduction) that results from splitting the data on a feature, then the following formula is applied [13]:

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)}((\frac{|S_V|}{|S|} \times Entropy(S_v)) \quad (4)$$

Such that ∑ denotes the summation of each one of the probability values of the attribute A. SV represents the S subset for which A has the value v, $|S_v|$ denotes the number of items in S_v. While, |S| represents the number of items in S.

3.3 Random forest classifier

It is an ensemble classification algorithm that includes numerous decision trees and discovers the classification results of every tree. The random forest classifier is typically fast. It can expand to many input feature variables and choose examples without pruning. In the training phase, the forest trees can be developed to maximal depth, and every tree is categorized separately. In the testing phase, the feature is appointed to every tree, and the forest chooses the class with maximum votes over all the trees using majority voting on average [6, 19].

3.4 Naïve bayes classifier

It is an easier probabilistic classification algorithm that depends on the Bayes rule with powerful independence presumption. A Naive Bayes classifier exhibits considerable performance under situations where the appearing words are distinct from one

another. The Naive Bayes classifier involves two phases, training and testing phrases. Both are based on text records represented by keywords (features) [11]. The two phases can be so summarized as follows: -

3.4.1 Training phase

Training the dataset starts with computing the number of records in each class and the total number of records in the dataset to find the prior probability of each class. NB classifier is learned with known keywords (features) that appear in the classes

1) Computation of the class conditional probability using the equation [11]:

$$P(\mathrm{x}|\omega_j) = P(x_1|\omega_j).P(x_2|\omega_j).\ \ldots.\ .P(x_d|\omega_j) = \prod_{k=1}^{d} P(x_k|\omega_j) \quad (5)$$

where ω_j is the class kind or class number.d is the total number of records. x_i is the feature vector of sample *i*.$P(\mathrm{x}|\omega_j)$ denotes how expected is it to notice the pattern x belongs to the class ω_j.

2) Computation of conditional probability (or likelihood) of every individual keyword in each text record of each class [20, 21]:

$$P(x_i|\omega_j) = \frac{N_{x_i,\omega_j}}{N\omega_j},\ i = 1, 2, \ldots., \mathrm{d} \quad (6)$$

whereN_{x_i,ω_j} is the number of times feature x_i appears in samples from class ω_j. $N\omega_j$ is the total count of all features in class ω_j.

3.4.2 Testing Phase

In this phase, the Naive Bayes classifier categorizes unknown text records in the testing set into one of the classes. This phase uses the calculated keywords probabilities of the training set from the training phase to classify the new testing records. The testing phase is implemented as follows:

1) Apply NB classifier to every textual record in the testing set formats
2) Get the keywords probabilities of the current test record from the training phase.
3) Suppose the keyword probability does not exist in the training stage. In that case, the algorithm applies the Laplacian smoothing technique to handle the zero-probability problem by adding one to every counter of zero appeared keywords.
4) Computing the general posterior probability using the following equation [11, 22]:

$$P(\omega_j|x_i) = \frac{P(x_i|\omega_j).P(\omega_j)}{P(x_i)} \quad (7)$$

5) After finding the posterior probabilities of the tested record for each class, the NB Classifier compares the results to find the largest probability, then assign the test record to the class with the maximum probability [20, 22].

3.4.3 There are two types of Naïve Bayes Classifier:

3.4.3.1 Multinomial nb model

By using a multinomial event model, every text record is expressed as a collection of word appearances on the record. The arrangement of keywords is not grasped. It produces the (bag of words) technique for representing records or documents. Using this model, the conditional probability of a text record x given a class ω can be computed by the product of the individual keyword's likelihoods in the corresponding class. The Term Frequency $tf(t,d)$ can be utilized to calculate the maximum likelihood of keywords depending on the training dataset to evaluate the class conditional probability as the following equation [23]:

$$P(x_i|\omega_j) = \frac{\sum tf(x_i,\ d \in \omega_j) + \propto}{\sum N_{d \in \omega_j} + \propto .\ V} \quad (8)$$

Where $\sum tf(x_i,\ d \in \omega_j) + \propto$ is The sum of raw term frequencies of word x_i from all documents in the training sample that belong to the class ω_j . $\sum N_{d \in \omega_j}$ is the sum of all term frequencies in the training dataset for class ω_j. $\propto$ is the smoothing parameter, where $\propto$=1 for Laplacian smoothing. V is the vocabulary size (the total words' number in the training dataset).

3.4.3.2 Multi–variate bernoulli NB model

In the Bernoulli NB model, every text record is expressed as a vector x of binary feature values that indicates which words appear and do not appear in the record. In addition, each word is represented as a Boolean value (i.e., 0 or 1) representing the appearance and the absence of that word. In this model, the keywords (features) are distinct binary variables, as well as Bernoulli

NB does not concern how many times the keyword appears in the textual record. Bernoulli's model is very common in categorizing tokens or keywords, as it has the advantage of clearly representing the word absence in the record or document. The Bernoulli trials can be written as [21]:

$$P(\mathrm{x}|\omega_j) = \prod_{i=1}^{m} P(x_i|\omega_j)^b . \left(1 - P(x_i|\omega_j)\right)^{(1-b)}, (\mathrm{b} \in 0, 1) \quad (9)$$

Let $p(x_i|\ \omega_j)$ represents the maximum-likelihood approximation that a specific token x_i happens in class ω_j and can be calculated as follows [21]:

$$P(x_i|\omega_j) = \frac{df(x_i,y)\ +1}{df_y\ +\ 2} \quad (10)$$

where $df(x_i, y)$ represents how many documents in the training set that includes the keyword xi and belongs to the class ω_j. df_y is the number of documents in the training dataset that belong to the class ω_j.+1 and +2 are the parameters of Laplace smoothing.

4. The proposed technique

The proposed technique classifies a product's data set into classes using various supervised classification algorithms such as K-Nearest Neighbor KNN, Decision Tree, Random Forest, Bernoulli model of a Naive Bayes classifier, and Multinomial model of the Naive Bayes classifier. The proposed technique combines machine learning classifiers with the feature extraction TF-IDF method to represent the keyword weights in the dataset records and improve classification accuracy. The proposed technique evaluates the classifier's performance for all machine learning algorithms by calculating accuracy, precision, recall, and f1-measure. The suggested technique compares the algorithm results and determines the best classifier used to classify the dataset records with higher performance measurement results.

The suggested technique splits the dataset records in every class into training and testing text records, aggregates them, and obtains the training and testing sets. Then, the proposed technique utilizes the training records to learn and train the classifier algorithms. Thereafter, the classifiers can categorize the testing records and assign them to their predicted categories, and the suggested technique calculates the evaluation measurements for each classifier.

The proposed technique uses a products dataset called GrammarAndProductReviews.csv, which includes nearly 65,500 records involving products, brands, and customer comments and reviews. The proposed technique utilizes the (categories) attribute of the dataset, which includes the keywords of products that the user searches to find them to classify the dataset depending on them. Each dataset record may have numerous keywords which represent the consumer search. Figure 1 shows the block diagram of the proposed technique framework.

4.1 Preprocessing the Dataset

First of all, the proposed technique applies several preprocessing steps on the dataset to remove any non-useful and irrelevant data, clean data, and prepare them to be input to the machine learning classifiers. The preprocessing steps can be explained as follows:

4.1.1 Tokenization

The proposed technique converts the (category) dataset attribute keywords in every record into tokens. Tokenization also removes any unwanted symbols and punctuation marks.

4.1.2 Stop Word Removal

The proposed technique removes and eliminates the stop words appear in the keywords in every dataset record, such as (his, her, the, at, in, is) utilizing a stop words list. The proposed technique compares each token with the list, and if the token appears there, then it will be removed from the dataset record. This process is iterated for all dataset records to remove their useless stop words.

4.1.3 Stemming

Stemming is a process that is applied to all the words of the dataset records to get their roots. For example, the stemmer removes the suffixes of the word (suppling) to be (supply), and the word (cleaners) will be returned to its root (clean).

4.1.4 Text Normalization

The text normalization can be applied to the stemmed keywords by uniforming the letter case of all words to lowercase. The same words (keywords) written in uppercase and lowercase will have the same meaning and can be considered to be the same words.

4.2 TF-IDF feature extraction

After performing text normalization on dataset record keywords, the proposed technique transforms each text keyword into its weight (TF-IDF value) to be a numeric weight that expresses the importance of the keyword in the record. Next, each text record is converted to a feature vector in the vector space model, which involves the TF-IDF weights that correspond to the text keywords in the record. After that, the proposed technique replaces the text keywords in the record with a feature vector of TF-IDF weights, and the dataset records will be a dataset of feature vectors of keyword weights. Term Frequency (TF) and Inverse Document Frequency (IDF) can be calculated using the following formulas [21]:

$$tf(term, doc) = \frac{frequency\ of\ term \text{ in } doc}{No.of\ terms\ \ in\ doc} \tag{11}$$

$$idf(term) = log\left(\frac{total\ no.of\ \ documents}{No.\ \ of\ docs\ that\ contain \text{ the } term}\right) \tag{12}$$

The proposed technique uses the preprocessed dataset with TF-IDF feature vector records as an input to the machine learning classifiers.

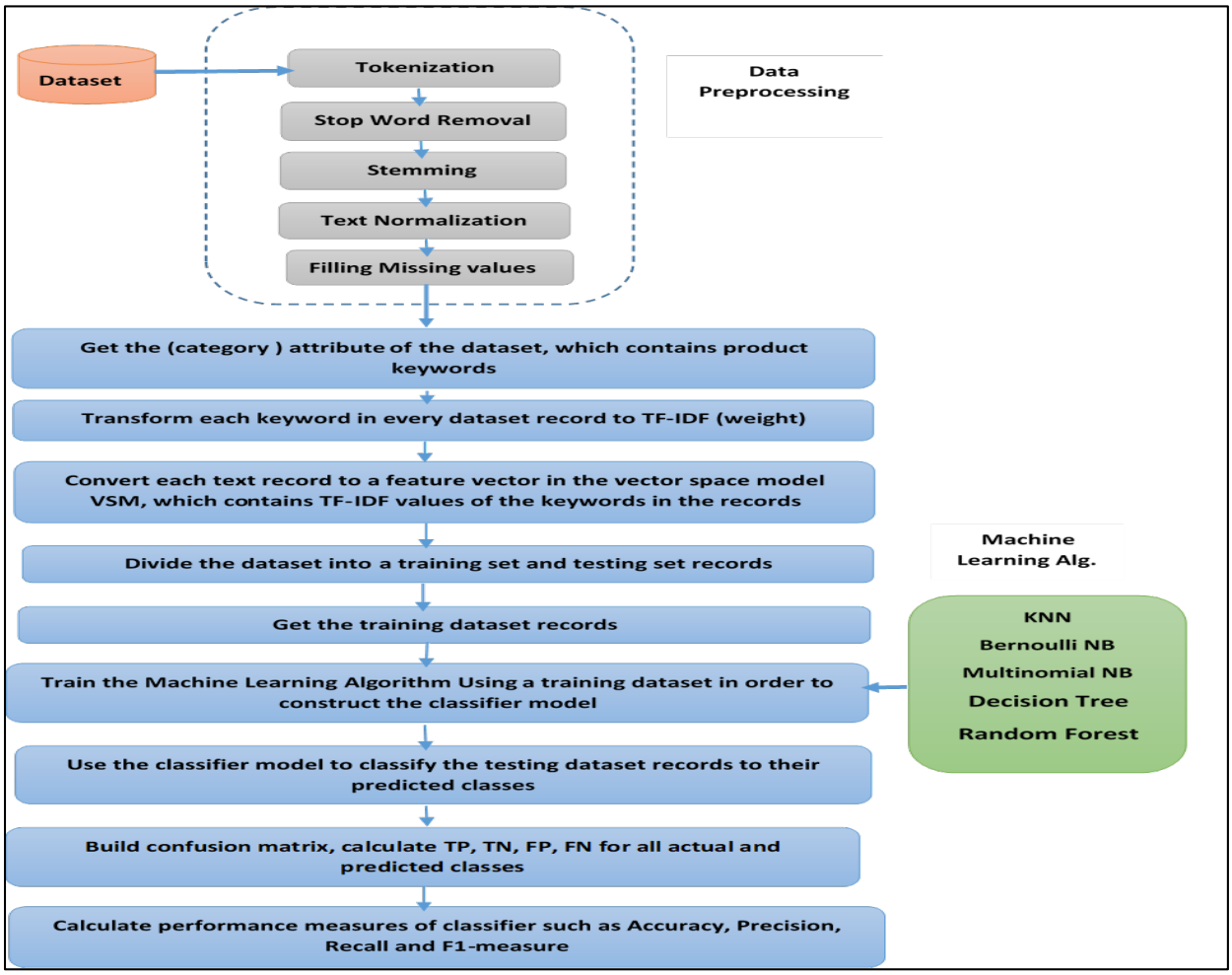

Figure 1: Block Diagram of the Proposed Technique Framework

4.3 Classification algorithms

The proposed technique divides the dataset records into two parts: 70% of the records represent the training dataset, and 30% of the records represent the testing. The proposed technique learns and trains the classifiers using the training set examples. The proposed technique utilizes several machine learning classifiers such as KNN, Decision Tree, Random Forest, and Naïve Bayes classifiers to classify the testing dataset records to their appropriate classes and calculate the evaluation metrics for each classifier to estimate the classifier performance. The proposed system decides the best classifier with the highest metrics that classify the dataset. Algorithms 1, 2, 3, and 4 show the customized KNN, Decision Tree, Random Forest, and Naïve Bayes classifier algorithms, respectively.

5. Experimental results

After testing each classifier with test dataset records which comprise 30% of the whole dataset, the technique evaluates the performance of the classifiers by measuring evaluation metrics such as accuracy, precision, recall, and f1-measure, and decides the best classifier that categories the data set into categories with higher measurement values. The proposed technique calculates the confusion matrix with TP, TN, FP, and FN for the classes of every classifier. Then it finds the evaluation measurements to assess the effectiveness and performance of each classifier algorithm.

The dataset records contain the category attribute, which involves the textual keywords that represent the product keywords. In addition, the dataset also contains the brand and manufacturer of products, URL of the products, customer reviews, customer recommendations, the city of the user who enters the product keywords query, and many other attributes. The proposed technique works on the category attribute containing the textual product keywords, classifies the dataset according to the product keywords using several machines learning classifiers, and determines the best classifier used to classify the textual dataset with higher evaluated metrics.

The proposed technique yields higher precision and accuracy results equal nearly to 0.99 in most used classifiers, which means that the utilized textual dataset has been processed effectively in terms of implementing several preprocessing steps such as tokenization, stop word removal, stemming and text normalization, to get cleaned and suitable data to be as input to classification algorithms. In addition, the higher precision and accuracy results indicate that the classifier models are correct and accurate enough to classify the dataset precisely, depending on the preprocessed data. For the KNN algorithm, the classifier obtains overall accuracy of 0.998372. The total classifier precision, recall, and f1-measure are 0.95, 0.98, and 0.96, respectively. Table 1 shows the precision, recall, and f1-measure values of all the 23 classes using the KNN classifier. As shown in this table, the KNN total Accuracy = 0.9983720, total Precision = 0.95, total Recall = 0.98 and total f1-measure = 0.96.

Table 1: The Results of the Precision, Recall, and f1-measure using the KNN technique

Class	Precision	Recall	F1-Measure
0	1.00	0.99	0.99
1	1.00	1.00	1.00
2	1.00	0.96	0.98
3	1.00	1.00	1.00
4	1.00	0.99	0.99
5	1.00	1.00	1.00
6	1.00	0.99	0.99
7	0.99	1.00	0.99
8	1.00	0.99	1.00
9	0.43	1.00	0.60
10	1.00	1.00	1.00
11	0.90	1.00	0.95
12	0.99	1.00	1.00
13	1.00	1.00	1.00
14	1.00	0.99	1.00
15	0.89	0.80	0.84
15	0.98	0.98	0.98
17	0.96	1.00	0.98
18	0.88	1.00	0.93
19	0.98	1.00	0.99
20	0.99	0.99	0.99
21	0.97	0.97	0.97
22	0.98	0.97	0.98
Total	0.95	0.98	0.96

In the Decision Tree algorithm, the overall algorithm accuracy is 0.999135. The total classifier precision, recall, and f1-measure are 0.99, 0.99, and 0.99. The algorithm obtains the following outcome for the random forest classification technique: 0.9995930, 0.99, 0.99, 0.99 values for classifier accuracy, precision, recall, and f1-measure, respectively. Using the Bernoulli model of the Naive Bayes algorithm, the classifier gets an accuracy value of 0.980566, while the other classifier metrics results are: precision of 0.86, recall of 0.80, and f1-measure 0.82. When implementing the multinomial naive Bayes classifier for testing the dataset records, the algorithm gains the following classification results: 0.982957 value for accuracy, 0.83 for precision, 0.86 for recall, and 0.84 for f1-measure. Tables 2, 3, 4, and 5 illustrate the detailed measurement outcomes for all the 23 classes for all classifiers. They show the accuracy, precision, recall, and f1-measure values for all the Decision Tree, Random Forest, Bernoulli Naive Bayes, and Multinomial Naive Bayes classifiers

Table 2: The Precision, Recall, f1-measure for each class using decision tree technique

Class	Precision	Recall	F1-Measure
0	0.99	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	0.98	0.99
8	1.00	0.99	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	0.86	0.92
12	0.99	0.99	0.99
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	0.88	1.00	0.93
16	0.97	0.98	0.98
17	1.00	0.96	0.98
18	0.95	1.00	0.97
19	0.99	1.00	0.99
20	1.00	1.00	1.00
21	0.96	0.97	0.97
22	1.00	0.98	0.99
Total	0.99	0.99	0.99

Table 3: The Precision, Recall, F1-measure for 23 classes using the Random Forest classifier

Class	Precision	Recall	F1-Measure
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	0.99	0.99
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	0.99	0.99	0.99
8	1.00	1.00	1.00
9	0.75	1.00	0.86
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	0.88	0.93
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
21	0.99	0.99	0.99
22	1.00	0.98	0.99
Total	0.99	0.99	0.99

Table 4: The Precision, Recall, F1-measure results for 23 classes using the Bernoulli model of Naïve Bayes classifier

Class	Precision	Recall	F1-Measure
0	0.81	0.98	0.89
1	0.99	1.00	0.99
2	1.00	0.75	0.86
3	0.99	0.98	0.99
4	0.88	0.91	0.89
5	0.84	0.99	0.91
6	1.00	0.86	0.92
7	0.93	1.00	0.97
8	0.95	0.99	0.97
9	0.00	0.00	0.00
10	1.00	1.00	1.00
11	0.91	0.64	0.75
12	0.84	0.65	0.73
13	1.00	1.00	1.00
14	1.00	0.99	1.00
15	0.00	0.00	0.00
16	1.00	0.61	0.76
17	0.71	0.58	0.64
18	1.00	0.67	0.80
19	0.97	0.94	0.95
20	0.96	0.92	0.94
21	0.97	0.96	0.96
22	0.94	0.92	0.93
Total	0.86	0.80	0.82

Table 5: The Precision, Recall, and F1-Measure for 23 Classes using the Multinomial Naïve Bayes Algorithm

Class	Precision	Recall	F1-Measure
0	0.84	0.96	0.89
1	1.00	0.99	0.99
2	0.91	0.76	0.83
3	1.00	0.98	0.99
4	0.96	0.95	0.95
5	0.85	1.00	0.92
6	1.00	0.98	0.99
7	0.90	1.00	0.95
8	0.96	1.00	0.98
9	0.00	0.00	0.00
10	0.99	1.00	0.99
11	0.71	0.94	0.81
12	0.49	0.96	0.64
13	1.00	1.00	1.00
14	1.00	0.99	1.00
15	0.00	0.00	0.00
16	0.96	0.70	0.81
17	0.82	0.95	0.88
18	0.88	1.00	0.93
19	0.99	0.92	0.95
20	0.97	0.91	0.94
21	1.00	0.83	0.91
22	0.95	0.95	0.95
Total	0.83	0.86	0.84

After comparing the evaluation metrics results, it is obvious that the Random Forest classifier is the best algorithm utilized in classifying data sets with the highest accuracy of 0.9995930, the highest precision, recall, and f1-measure of 0.99,0.99 and 0.99,

respectively. On the other hand, the Decision Tree represents the second-best classifier algorithm with an accuracy of 0.999135, which is lower slightly than the Random Forest algorithm. Table (6) lists the classifier algorithms and their results sorted in descending order from the classifiers with the highest accuracy, precision, recall, and f1-measure value of the lowest values to determine the best and ideal classifiers classify the dataset with the highest results.

Table 6: Summary of the Classifiers with the Highest Accuracy, Precision, Recall, and F1-Measure Results, Sorted in Descending Order

Number	Classifier	Accuracy	Precision	Recall	F1-Measure
1	Random Forest	0.9995930	0.99	0.99	0.99
2	Decision Tree	0.9991351	0.99	0.99	0.99
3	KNN	0.9983720	0.95	0.98	0.96
4	Multinomial NB	0.9829577	0.83	0.86	0.84
5	Bernoulli NB	0.9805667	0.86	0.80	0.82

6. Conclusion

Text classification is the process of appointing predetermined classes to text dataset records or documents, and it may supply views of document groups. Rather than classifying text documents manually, various machine learning techniques have been performed automatically to categorize documents depending on the training documents set. This paper introduces an approach that has employed several supervised machine learning techniques based TF-IDF feature extraction method to automatically classify a product textual keywords data set of nearly 65,500 records into individual classes. The suggested technique implements several classifiers such as KNN, Random Forest, Decision tree, Bernoulli, and Multinomial models of Naive Bayes classifiers. The proposed method utilizes TF-IDF as a vector space model to get more accurate classifier outcomes. The proposed technique produces higher Accuracy, Recall, Precision, and F1-measure evaluation metrics for all classifiers. After comparing the classifier's evaluation results, the suggested technique shows that the Random Forest algorithm is the best and ideal classifier utilized to categorize the dataset with the highest Accuracy value of 0.9995930.

Algorithm (1): Customized K-Nearest Neighbor Classifier.
Input: The dataset of feature vectors as TF-IDF keywords weights
Output: classified or predicted testing feature vector set
Begin
Step 1: Load the weight vector data set (training and testing sets).
Step 2: Choose the appropriate k value.
Step3: For the training phase, store the training set feature vectors weights with their related class labels
Step 4: In the testing phase from the testing dataset, get the next testing feature vector sample.
Step 5: Calculate the Euclidian Distance between all feature vectors of the training set with the tested feature sector sample.
Step 6: Distances are sorted in ascending order in a list.
Step7: Top k tuples are selected from the sorted list; they represent the k-nearest neighbors' closest distances to the tested sample.
Step 8: Identify the predicted class using the majority vote of the k- nearest neighbor's classes, i.e., get the most frequent class from the top k rows of the sorted distances list of nearest neighbors.
Step 9: Return the predicted class of tested feature weights vector.
Step 10: Get the next testing feature vector sample from the testing set until it becomes empty, and return to step 5. If the testing set is empty, then Exit.
End

Algorithm (2): Customized DT Classifier.
Input: Training set records S, Features set F ($f_{1, ..}$, f_n)
Output: DT
Begin
Step 1: Create the root node in the tree.
Step 2: If all the records examples in the training set S are in the same class, then
Begin DT= single node tree with class C Return DT
End
Step3: Choose the feature f that has the largest (Information Gain) value
Step4: For every feature f from the feature field:
BeginAdd a new branch that corresponds to this better feature f. Add a near node that saves all the examples of records with values for this feature.
End
Step5: If the node stores example records that belong to only one class C, then
Begin this node becomes a leaf node else go to step 6.
End
Step6: Below the node's branch, build a new subtree, and go recursively to step 3
Step7: Return DT
End

Algorithm (3): Customized Random Forest classifier
Input: Training set S, Features F
Output: RF
Begin
Step1: To produce c classifiers,
for j = 1 to c do
Randomly sample the training data S with replacing to generate Sj
Step2: Create a root node Nj, that contains Dj
Step3: call Construct Tree (Nj)
Construct Tree (N)

Algorithm (4): Customized Naive Bayes classifier
Input: training products dataset
Output: tested dataset classified using NB classifier
Begin
for each class in the training dataset
Begin
Calculate the prior probability of each class using Equation No. (5)
Calculate the likelihood of every keyword in each record using Equation No. (6).
end
For each record in the testing, set do
Begin
for each class c_i in the training set do
Begin
Calculate the posterior probability of the record using Equation No. (7).
End
Compare the posterior probability results of the test records in all classes to find the largest probability.
Assign the record to the maximum probability class
End
Return the classified tested dataset.
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] A. Patra , D. Singh, A survey report on text classification with different term weighing methods and comparison between classification algorithms, Int. J. Comput. Appl., 75 (2013)14-18. http://dx.doi.org/10.5120/13122-0472

[2] D. Kalita, Supervised and unsupervised document classification-a survey, Int. J. Comput. Sci. Inf. Technol., 6 (2015)1971-1974.

[3] R. Jindal, R. Malhotra , A. Jain, Techniques for text classification: literature review and current trends, Webology, 12 (2015) 1-28.

[4] M. Mowafy, A. Rezk ,H. M.El-bakry, An efficient classification model for unstructured text document, American J. Comput. Sci. Inf. Technol., 6 (2018)1-10. http://dx.doi.org/10.21767/2349-3917.100016

[5] A. Patra , D. Singh, Neural network approach for text classification using relevance factor as term weighing method , Int. J. Comput. Appl., 68 (2013)37- 41. http://dx.doi.org/10.5120/11674-7301

[6] V. Korde, C. N. Mahender, Text classification and classifiers: a survey, Int. J. Artif. Intell. Appl., 3 (2012)85-99. https://doi.org/10.5121/ijaia.2012.3208

[7] W. Hadi, M. A. H. Eljinini , S. Alhawari, The automated arabic text categorization using SVM and KNN, knowledge management and innovation: a business competitive edge perspective, 2 (2010) 757-763.

[8] A. Bilski , A review of artificial intelligence algorithms in document classification, Int. J. Electron. Telecommun., 57 (2011) 263–270 . https://doi.org/10.2478/v10177-011-0035-6

[9] M. Thangaraj , M. Sivakami, Text classification techniques: a literature review, Interdiscip. J. Inf. Knowl. Manag., 13 (2018)117-135 . https://doi.org/10.28945/4066

[10] M. Manjotho, T. J. S. Khanzada, L. A. Thebo , A. A. Manjotho, Improving performance of mobile SMS classification using TF-IDF & multinational naïve Bayes classifier, Eng. Sci. Technol. Int. Res., 2 (2018)26-32.

[11] M. Abbas, K. Ali, A. Jamali, S. Memon , A. Ahmed, Multinomial naive Bayes classification model for sentiment analysis, Int. J. Comput. Sci. Netw. Secur., 19 (2019) 62-67. http://dx.doi.org/10.13140/RG.2.2.30021.40169

[12] W. M.U. Noormanshah, P. N.E. Nohuddin , Z. Zainol, Document categorization using decision tree: preliminary study, Int. J. Eng. Technol., 7 (2018) 437- 440. http://dx.doi.org/10.14419/ijet.v7i4.34.26907

[13] D. Singh, S. Malhotra, Intra news category classification using n-gram tf idf features and decision tree classifier, Int. J. Sci. Adv. Res. Technol., 4 (2018) 508-514.

[14] B. Trstenjak, S. Mikac , D. Donko, KNN with TF-IDF based framework for text categorization, Proc. Eng., 69 (2014) 1356 -1364. https://doi.org/10.1016/j.proeng.2014.03.129

[15] S. A. Nasser , I. A. Hashim ,W. H. Ali, Visual depression diagnosis from face based on various classification algorithms , Eng. Technol. J., 38 (2020) 1717-1729. https://doi.org/10.30684/etj.v38i11A.1714

[16] N. T. Mahmood, M. H. Al-Muifraje , S. K. Salih , T. R. Saeed, Pattern recognition of composite motions based on emg signal via machine learning , Eng. Technol. J., 39 (2021) 295-305. https://doi.org/10.30684/etj.v39i2A.1743

[17] M. A. jabbar, B. L. Deekshatulua , P. Chandra, Classification of heart disease using k- nearest neighbor and genetic algorithm , Proc. Technol., 10 (2013) 85-94. https://doi.org/10.1016/j.protcy.2013.12.340

[18] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, Using KNN model-based approach for automatic text categorization, Soft. Comput., 10 (2006) 423–430. https://doi.org/10.1007/s00500-005-0503-y

[19] N. N. A. Sjarif, N. F. M. Azmi, S. Chuprat, H. M. Sarkan, Y. Yahya, S. M. Sam, SMS Spam message detection using term frequency-inverse document frequency and random forest algorithm, Proc. Comput. Sci., 161 (2019) 509-515. https://doi.org/10.1016/j.procs.2019.11.150

[20] H. Shimodaira, Text classification using naive Bayes, (2015).

[21] S. Raschka, Naive Bayes and text classification I introduction and theory, (2014).

[22] N. C. Le, P. W. C. Prasad, A. Alsadoon, L. Pham , A. Elchouemi, Text classification: naïve Bayes classifier with sentiment lexicon, Int. J. Comput. Sci., 46 (2019)141-148.

[23] S. Xu, Y . Li, Z.Wang, Bayesian multinomial naïve Bayes classifier to text classification, Lect. Notes. Electr. Eng., 448 (2017) 347–352. https://doi.org/10.1007/978-981-10-5041-1_57