 Acta Univ. Sapientiae Informatica **15,** 1 (2023) 91–110

DOI: 10.2478/ausi-2023-0008

# Textual outlier detection with an unsupervised method using text similarity and density peak

Mahnaz TALEB SERESHKI
Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran
email: M.talebsereshki@edu.ikiu.ac.ir

Morteza MOHAMMADI ZANJIREH[a]
Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran
email: Zanjireh@eng.ikiu.ac.ir

Mahdi BAHAGHIGHAT
Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran
email: Bahaghighat@eng.ikiu.ac.ir

[a]Corresponding author

**Abstract.** Text mining is an intriguing area of research, considering there is an abundance of text across the Internet and in social medias. Nevertheless outliers pose a challenge for textual data processing. The ability to identify this sort of irrelevant input is consequently crucial in developing high-performance models. In this paper, a novel unsupervised method for identifying outliers in text data is proposed. In order to spot outliers, we concentrate on the degree of similarity between any two documents and the density of related documents that might support integrated clustering throughout processing. To compare the effectiveness of our proposed approach with alternative classification techniques, we performed a number of experiments on a real dataset. Experimental findings demonstrate that the suggested model can obtain accuracy greater than 98% and performs better than the other existing algorithms.

**Key words and phrases:** outlier detection, text data processing, unsupervised learning

## 1 Introduction

Countless specialists in data analysis are required for the precise processing and interpretation of data [21, 8, 33, 13, 37, 3]. In contrast, the influence of the Internet of Things (IoT) and social media has led to an enormous rise in the accumulation of data [13], [34]. Generally, data generation is prone to noise and unwanted changes. Since there is a lot of dirty data due to misinformation, disinformation, or bugs in data gathering, storage or call procedures, data cleansing can aid data analysts in achieving their objectives to create high accuracy models. Finding outliers is a proactive data-cleaning method. It is defined as an algorithm which tries to probe abnormal data [23]. Abnormal data represents information that deviates from the dataset's predominant patterns [22, 42]. In certain circumstances, system designers target finding outliers to study why these abnormal data are available in the system [38]. The significance of identifying outliers is demonstrated in credit fraud prevention and intrusion detection in computer networks [41, 24]. Because these sorts of complicated data need a lot of processing, spotting outliers in data can assist data scientists in improving the performance of their models [13, 24, 11]. Text data outliers can take on a variety of forms, and it might be challenging to identify them in this application.

On the identification of outliers in texts, numerous researchers have concentrated. As consequence, multiple aspects of the text's qualities are taken into account. A few, for instance, converted language into numbers and applied methods that are suitable for numerical data. Additionally, a number of research efforts exploited restricted phrases, such as document titles, to seek out patterns in datasets as well as recognize outliers. Plenty of studies have been conducted on dynamic datasets, such as social media, and the detection of anomalous data in such circumstances. It additionally addresses how significant knowledge is before any analysis ([23, 11, 20, 2, 30, 12, 35, 16]).

In this paper, our goal is to identify documents that do not follow the primary patterns of datasets in order to develop a high-performance text processing model from the total clusters that can be flexible in various situations, appropriate for text features, and implementation-unrestricted.

The remainder of the paper is structured as follows. The different kinds of methods for identifying outliers are discussed in Section 2. The solution we propose is elaborated in Section 3. Section 4 highlights our findings under various circumstances. Ultimately, Section 5 serves as the conclusion.

## 2 Related works

In general, there are several distinct types of outlier detection techniques. The classification-based algorithms are the subject of the first. Data in this situation ought to be labeled and utilized to train classification systems [17]. In contrast, the input to a clustering algorithm is not labeled, and the algorithm learns by solving problems similar to those it will face in the future as part of a training schedule. This makes a clustering algorithm an unsupervised model [28, 18].

Outlier detection is also possible with the clustering technique. Outlier detection based on clustering that comes following aims detection of "abnormal data". These algorithms identify outliers—those data that do not follow the typical data's patterns—as the data. Nearest neighbor is taken into account by several algorithms. The nearest neighbor plays a key role in these algorithms. Here, the term "abnormal data" refers to data that do not resemble their neighbors in any way [17, 25].

Other methods compute the probability of data being in a particular area employing probabilities and statistical models; if data exist in a region where this likelihood is low, the data is considered an outlier [17, 6, 5].

Subsequently, a few techniques are used for datasets that are dispersed throughout several platforms, like big data. These algorithms are made to look for outliers by taking into account the patterns of all the data in all systems [41, 31].

The remaining part of this section represents two well-known algorithms that we tested. One of them is a member of the clustering group, whereas the other one views the other as nearby neighbors.

### 2.1 Density-based spatial clustering of applications with noise (DBSCAN)

DBSCAN is one of the density-based clustering algorithms [10]. In this algorithm, two parameters are inputs $(\mu, \varepsilon)$, and points or data are categorized as core points, reachable points, and outliers. They are defined as follows:

- Core points are the points with at least $\mu$ neighbors in the border of $\varepsilon$ from itself
- Reachable points are those points which are not core points but are located on the border of $\varepsilon$ from one or more than one core point
- Outliers are those points that are not core or reachable points

Figure 1 demonstrates the differences between these kinds of points.

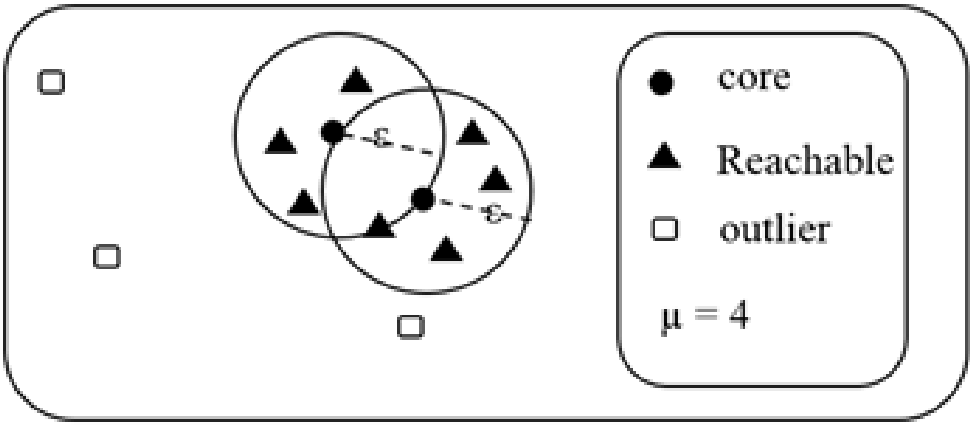


Figure 1: Core, reachable, and outlier points in DBSCAN

## 2.2 The improved Local Outlier Factor (LOF)

The goal of LOF which is based on the local density and nearest neighbors, is computing the local outlier factor which is the outliers' determination [40]. For computing the LOF of each point, calculating the reachability distance between the points and their k nearest neighbors, and also the Local Reachability Density (LRD) of the points are necessary. The reachability distance between two points $(p, q)$ is defined as [12]:

$$reach\_dist(p, q) = max(k - distance(q), d(p, q)) \tag{1}$$

In Equation (1), $k - distance(q)$ is the distance between $q$ and its k'th nearest neighbor, and $d(p, q)$ is the distance between $p$ and $q$. The concept of reachability distance is shown in Figure 2.

And local reachability density of each point is defined as follows [12]:

$$LRDk(p) = K\frac{\sum_{q \in knn(p)} reach - dist(p.q)}{||k - neighborhood||} \tag{2}$$

In Equation 2, $q$ is the neighbor of $p$, $reach - dist(p, q)$ is the reachability distance between $p$ and $q$, and k-neighborhood is the number of $p$'s neighbors. Finally, the Local Outlier Factor of each point is defined as [12]:

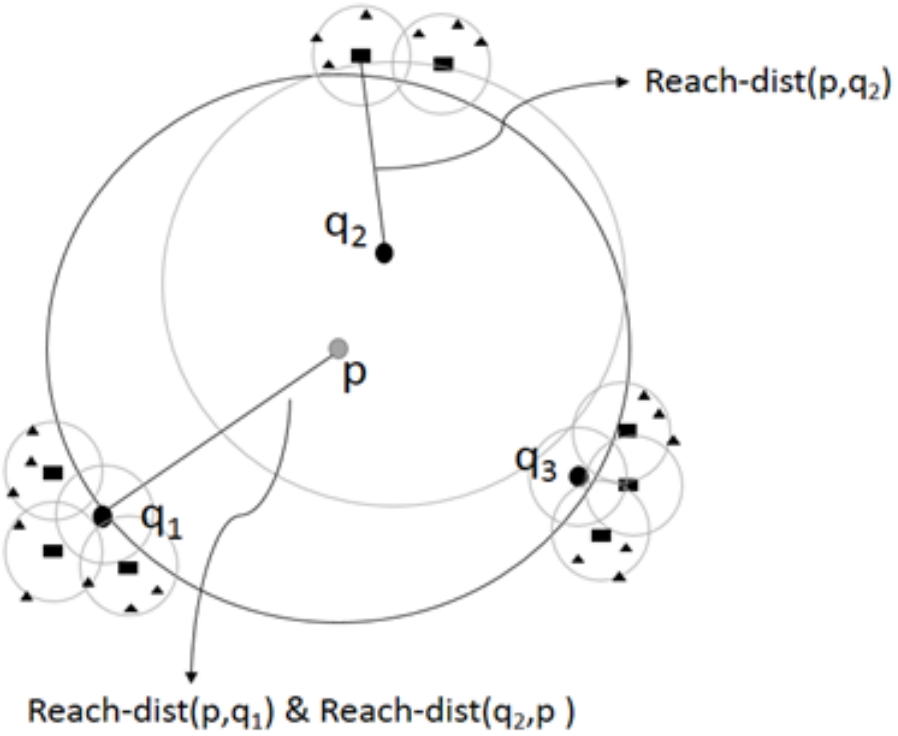


Figure 2: Reachability distance between two points [37]

$$\mathrm{LOFk}(p) = K\frac{\sum_{q\in knn(p)} \frac{\mathrm{LRD}(q)}{\mathrm{LRD}(p)}}{||k-neighborhood||} \tag{3}$$

In Equation 3, $q$ is the neighbor of $p$, $\mathrm{LRD}(p)$ and $\mathrm{LRD}(q)$ are the local reachability density of $p$ and local reachability density of $q$ respectively. Furthermore, k-neighborhood is the number of $p$'s neighbors. Whatever LOF is higher, the point is more abnormal.

## 3 Methodology

As we present in Section 2, the clustering algorithm is one of the groups of outlier detection, and this kind of algorithm can help to find abnormal data and the patterns of datasets. As mentioned before, the center of each cluster

is the point with a higher density than its neighbors and more different from the points with higher densities. In this case, the measurement of similarity is the distance between two points, but we prepare the algorithm for text data, therefore, cosine similarity is used. As a result, we propound local density for each document, denoting the number of text data or documents that are similar to this data. For this purpose, threshold and local density are considered as follows:

$$\rho_i = \sum_j X(t_i.t_j),\ X(a,b) = \begin{cases} 1, & \text{if } Similarity(a,b) > threshold \\ 0, & \text{if } Similarity(a,b) < threshold \end{cases} \tag{4}$$

In Equation 4, $\rho_i$ is the local density of document $i$, and $j$ is the number of the documents in a dataset, and $t_j$ is an $i_{th}$ document in the dataset. Figure 3 demonstrates computing $\rho_i$.

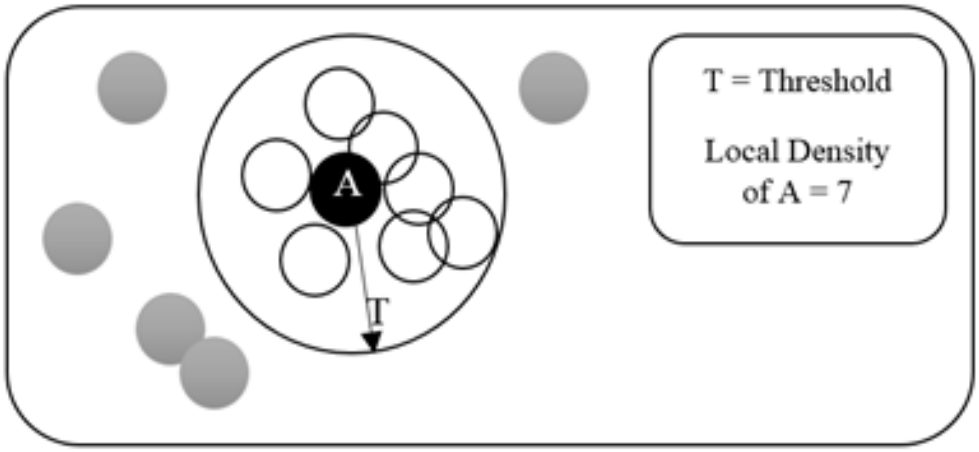


Figure 3: The local density of a document

$\delta_i$ is the similarity between document $i$ and the most similar document with higher local density. For example, as we can see in Figure 4, $\delta$ for the black document is the similarity between the gray document and itself.

Eventually, we suggest outliers that are more different from the behavior of central data. In other words, outliers are texts with less $\rho_i$ and fewer $\delta_i$.

Considering the purpose of finding outliers in text data and density peak, the rest of this section explains our new method. First of all, the similarity of each pair of texts is calculated and saved in a matrix (See Figure 5).

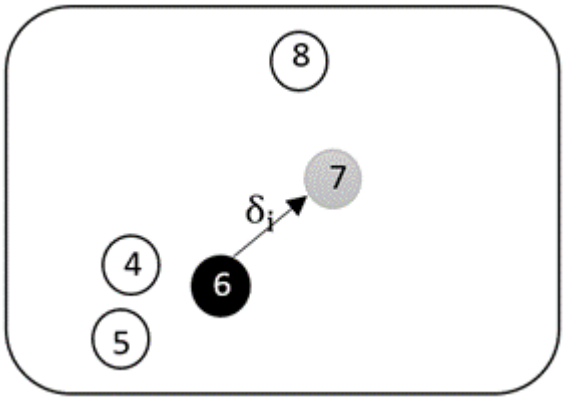


Figure 4: The computing of $\delta_i$

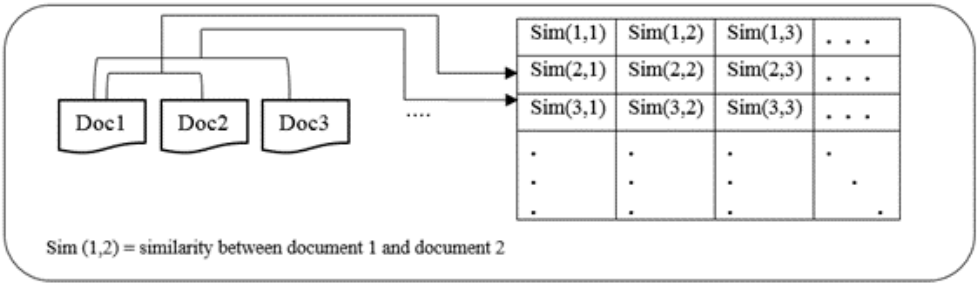


Figure 5: The computing of similarity matrix

Then a threshold should be set for the similarity between documents. Now for each document, $\rho_i$ is computed and saved in the $\rho$ array. Afterward, $\delta_i$ should be recognized for all documents and saved in the $\delta$ array. Finally, outliers are searched as the $\rho_i$ and $\delta_i$ are lower than the thresholds. We can see the steps of the algorithm in Algorithm 1.

**Algorithm 1**
**Input**: documents, $TS, Td, TS1$
**Output**: Outliers
1. Calculate the similarity index
2. Set the $\rho$ array
3. Set the $\delta$ array
4. Print the number of documents in which the $\rho_i < T_d$ and $\delta_i < TS1$ as outliers

### 3.1 Dataset

BBC dataset is utilised in our experiment to evaluate the performance of our suggested method for outlier spotting in the text. Different scenarios are used to test each dataset. First, several outlier identification methods prepare them for entrance into categorization algorithms. In these situations, data identified using our suggested approach, LOF, and DBSCAN is cleaned; otherwise, the datasets remain untouched. The accuracy of four distinct categorization methods is then calculated, and they are contrasted. These tests are designed to find out how our strategy of eliminating outliers affects classification algorithm performance.

### 3.2 Classification algorithms

In our experiments, four different classification algorithms (K Nearest Neighbor, Decision Tree, Random Forest, and Naïve Bayes) are implemented to compare different situations. Furthermore, Term Frequency (TF) [27], Term Frequency-Inverse Document Frequency(TF-IDF) [36], and 3-gram [9] are used for the inputs of these algorithms. We briefly describe these four algorithms in the rest of this section.

#### 3.2.1 K-nearest neighbor (KNN)

The k nearest neighbor classifier is a classification algorithm based on the distance between the input sample and the training samples. The algorithm will find the k objects which are closest to the input data, and the value of the input will be predicted according to the values of these neighbors. For instance, $xi$ is input, and $(x_{i_1}, xi_2, ..., x_{i_p})$ are its features. The Euclidean distance between $x_i$ and $x_l$ is computed as:

$$d(x_i, x_l) = \sqrt{(x_{i_1} - x_{l_1})^2 + (x_{i_2} - x_{l_2})^2 + ... + (x_{i_2} - x_{l_2})^2} \tag{5}$$

This distance is computed between $x_i$ and every training sample to find the nearest neighbors and the value of $x_i$ is considered regarding its $K$ nearest neighbors [19, 29], as shown in Figure 6.

#### 3.2.2 Decision Tree (DT)

One of the flowchart-like structure methods is a decision tree. It makes a tree based on training data when new data enters into the training model which is similar to the yes and no question game, predicts the input value. Each tree

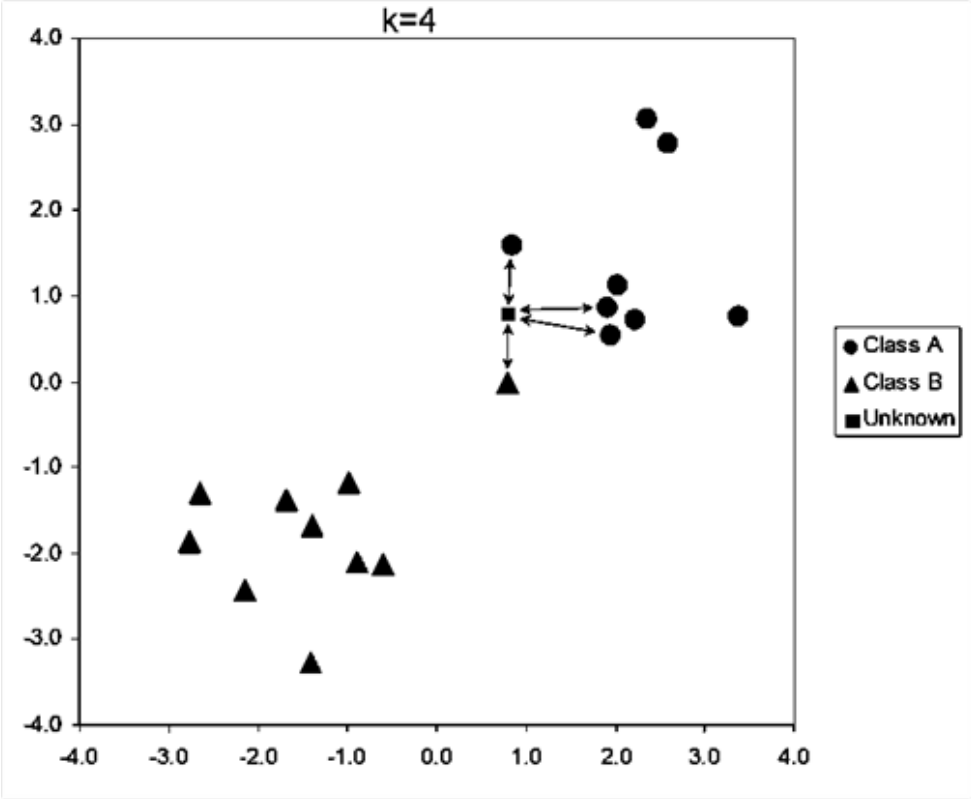


Figure 6: The vote for the unknown sample values recognition [26]

composes of nodes and branches. The nodes show features of classes, and they can get values by using branches. Figure 7 represents the decision tree as an example. Decision trees have found many fields of implementation due to their simple analysis [32].

### 3.2.3 Random Forest (RF)

Random Forest is an ensemble learning method for classification, regression, and other learning that builds a large number of decision trees during training. This algorithm makes a number of random decision trees with different properties, then the value of new data is decided by the voting of these trees [15]. Figure 8 illustrates the logic of the random forest algorithm.

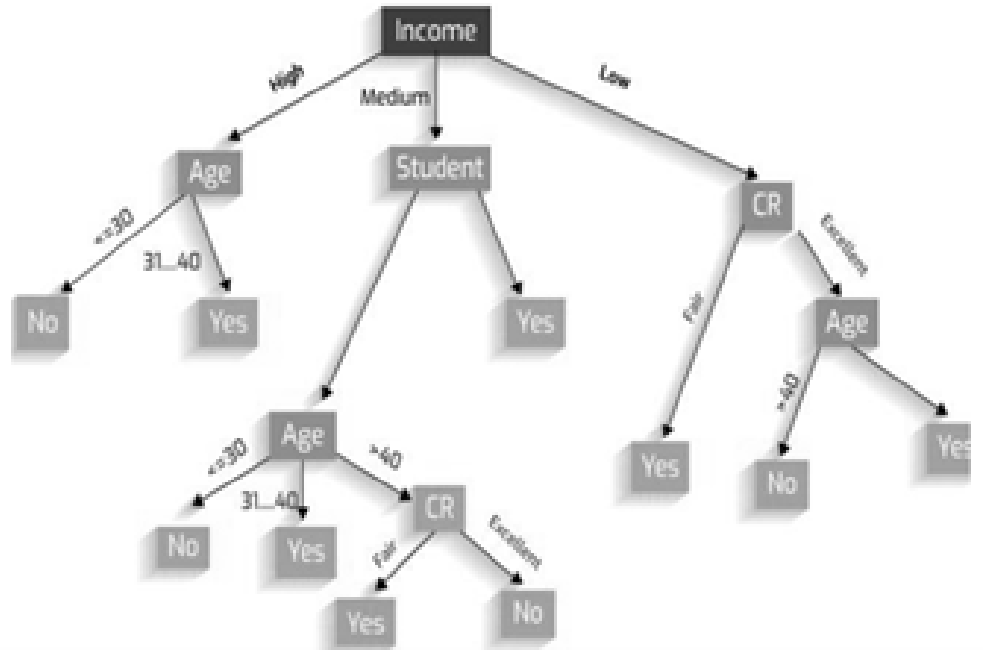


Figure 7: The example of a decision tree [7]

### 3.2.4 Naïve Bayes

Despite the simple design and assumptions, Naïve Bayes classifiers have worked very well in many complex real-world situations. Naïve Bayes is a probability model based on Bayes' theorem which is defined as:

$$P(A|B) = \frac{p(B|A).p(A)}{p(B)} \tag{6}$$

This algorithm calculates the probability of each input feature in each class by considering the training data and selects the most likely class as the value of the input sample. In this method, $X$ is the input, and $(x_1, x_2, \ldots, x_i)$ are its features. The values or classes are defined as $(C_1, C_2, \ldots, C_i)$. Now for each input, the probability of each class for the input is predicted by Equation 7 [14].

$$P(C_i|X) = \frac{p(X|C_i).p(Ci)}{p(X)} \tag{7}$$

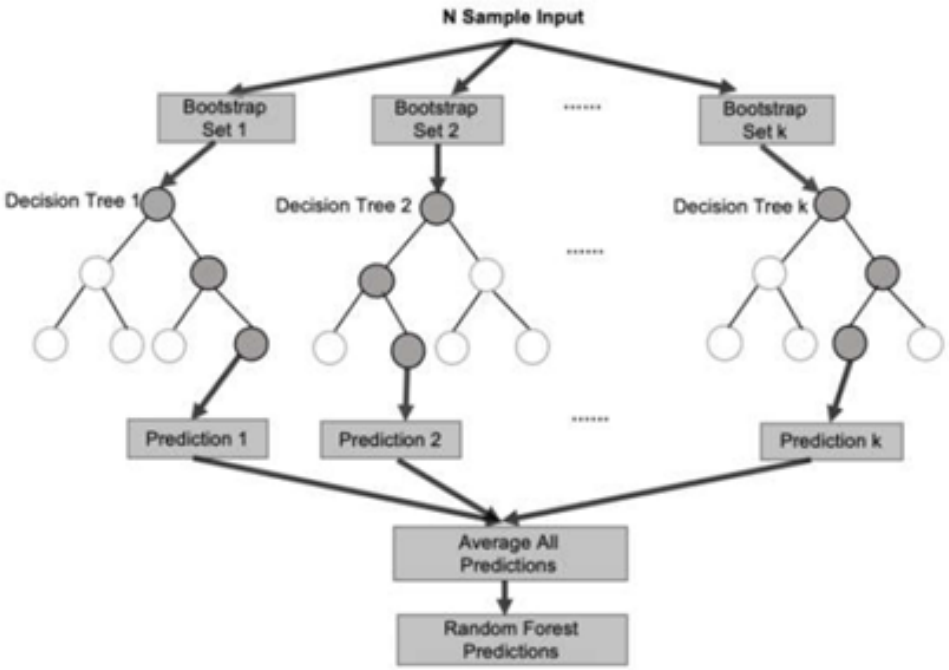


Figure 8: The process of decision in the Random Forest [39]

That $p(X|C_i)$ is calculated as:

$$P(X|C_i) = p(X|C_i) = \prod_{k=1}^{n} p(x_k|C_i) = P(x_1|C_i).P(X_2|C_i)...P(x_n|C_i) \quad (8)$$

## 4 Experimental results

According to Table 1, the proposed model was implemented using the Python programming language. To implement the models, two parts of the dataset were used for model training and evaluation. There is the 80% training dataset and the 20% evaluation dataset.

| | |
|---|---|
| OS | Windows 10-64bit |
| CPU | Intel(R) Core (TM)i7-7700HQ 2.8GHz |
| GPU | NVIDIA Tesla P100 |
| RAM | 16 GB |
| Programming language | Python 3.9 |
| Software libraries | TensorFlow, Keras, OpenCV, and Scikit-Learn |

Table 1: Hardware & software environments deployed in this study

### 4.1 Evaluation metrics for classification problems

Evaluation of any machine learning model's performance is the most crucial step in model construction. So, the question of how to evaluate a machine learning model's performance arises. Machine learning tasks are connected to evaluation measures. Regression and classification tasks each have their own metrics. Before we put our model into production on untested data, we should be able to increase its overall predictive power by evaluating its performance using several criteria. When a machine learning model is applied to unexplored data, failing to properly evaluate it using a variety of assessment measures and relying simply on accuracy can result in inaccurate predictions. Accuracy, confusion matrix, precision, recall, F1-score, sensitivity, specificity, and AUC are major performance measures for classification problems that are most frequently employed. Following equations represent these metrics:

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{9}$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{10}$$

$$\mathrm{F1\text{-}score} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{11}$$

$$\mathrm{Sensitivity} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{12}$$

$$\mathrm{Specificity} = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}} \tag{13}$$

Instances where the model accurately predicts a positive class are referred to as TPs (True Positives). This is taken to be true since the input really corresponds

to the positive class that the model predicted would exist. FP (False Positive) is the term for when a positive class is falsely predicted by a model, even if it can be perceived as the model doing so. FN (False Negative) is the term used when a model predicts a negative class wrongly, which is untrue but can be read as a negative class being predicted by the model. When the model properly predicts a negative class, this is known as a TN (True Negative), and is taken to be true because the input really corresponds to the predicted negative class.

Classification accuracy is the most often used performance metric for evaluating classification models. Due to the fact that it can be stated as a single number that encapsulates the model's capabilities and is straightforward to compute and comprehend, it is widely used. Therefore, in our simulations, accuracy was the performance indicator we used.

### 4.2 Simulation results

As can be seen in Table 2, all algorithms, including Decision Tree, Random Forest, and Naïve Bayes, performed better once outliers were removed using our method. Therefore, our approach significantly impacts when used as a pre-processing algorithm during text processing. Results demonstrate that our strategy improves the accuracy of the Nave Bayes classifier by more than 98%.

Table 3 compares the accuracy of our method with DBSCAN, and LOF for Random Forest algorithm. It constructs a forest using a collection of decision trees. Random Forests produce uncorrelated decision trees and execute feature selection implicitly. To accomplish, it constructs each decision tree using a random collection of features. This makes it a great model for working with data that has a lot of different properties. The results show that when Random Forests were used, our method's accuracy increased from 90% to 92.5%, despite the fact that Random Forests are not much influenced by outliers. This highlights how effective our approach is at detecting outliers.

The performance of our suggested detection of outliers technique will then be assessed when used for The K-Nearest Neighbor (KNN). The KNN machine learning technique is flexible. It is employed in a variety of contexts, including handwriting recognition, picture recognition, and video recognition. In a wide range of prediction problems, it can achieve high accuracy. KNN is a method that is used to learn an unknown function with the appropriate precision, and accuracy. It is based on the local minimum of the target function. The algorithm also determines a parameter's range or distance from an unknown input as well as its surroundings. Based on the "information gain" theory, the

| Algorithm | Decision Tree | | |
|---|---|---|---|
| Outlier detection | | Without deleting outliers | Our method |
| Accuracy | TF | 0.8071 | 0.8669 |
| | TF_IDF | 0.8071 | 0.8394 |
| | N-gram (n=3) | 0.7533 | 0.7114 |
| Algorithm | Random Forest | | |
| Outlier detection | | Without deleting outliers | Our method |
| Accuracy | TF | 0.8917 | 0.9149 |
| | TF_IDF | 0.9058 | 0.9278 |
| | N-gram (n=3) | 0.8466 | 0.8522 |
| Algorithm | Naïve Bayes | | |
| Outlier detection | | Without deleting outliers | Our method |
| Accuracy | TF | 0.9596 | 0.9862 |
| | TF_IDF | 0.9686 | 0.9816 |
| | N-gram (n=3) | 0.9327 | 0.9633 |

Table 2: The accuracy of Decision Tree, Random Forest, and Naïve Bayes algorithms on the BBC dataset

| Technique | With outliers | LOF (k=7) | DBSCAN (3,2) | Proposed |
|---|---|---|---|---|
| TF | 0.8917 | 0.9197 | 0.9548 | 0.9149 |
| TF IDF | 0.9058 | 0.9217 | 0.9031 | 0.9278 |
| N-gram | 0.8466 | 0.8235 | 0.8463 | 0.8522 |

Table 3: Comparing the accuracy of our method with DBSCAN, and LOF for Random Forest algorithm

algorithm determines which method is best suited to forecast an unknowable value. Figures 9, 10, and 11 demonstrate that our method along with KNN can get an accuracy more than 96% for K=10 and TF-IDF. It strongly outperforms other existing approaches presented in these figures in particular when an outlier detector was not implemented.

Using a k-fold cross-validation approach, we expanded our experimental findings and verified the effectiveness of our model. Our dataset was separated into k subsets ($k = 5$), the model was trained on k-1 subsets, and it was then assessed on the final subset. A new subset was tested in each of the k iterations of this process which was repeated. The performance of the model was then estimated by averaging the findings. The outcomes of our tests revealed that,

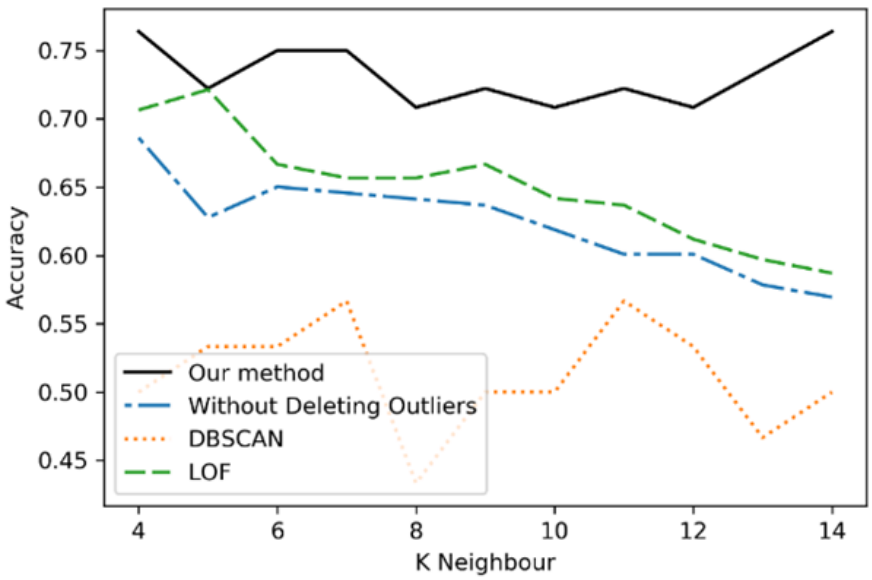


Figure 9: The accuracy of KNN using TF and k between 4 and 14 on the BBC dataset

when tested using cross-validation, our model had somewhat enhanced the performance to reach 98.92% accuracy.

## 5 Conclusion

Today, many applications leverage data mining and machine learning approaches ([1, 4, 11]). Given the fierce competition to obtain ever-increasing accuracy, any beneficial pre-processing procedures, such as outlier detection, would be taken into consideration in real systems. Additionally, text features indicate that processing this kind of data is challenging. Nevertheless, processing can be made smoother by spotting anomalous data. In this research, we have presented a unique method for identifying anomalous data in text datasets. The characteristics of the texts that we uncovered did not match those in the clusters' center. We created the density-based method in keeping with this idea. Then, we conducted a number of experiments to evaluate how well our approach performed compared to two popular outlier detectors

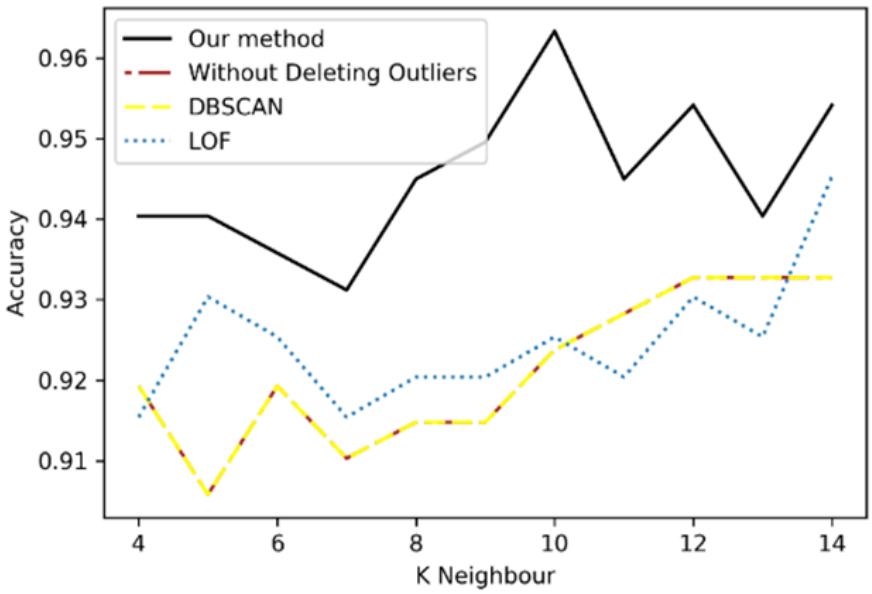


Figure 10: The accuracy of KNN on the BBC dataset with k between 4 to 14 with TF-IDF

(LOF and DBSCAN). For our tests, we used the BBC dataset. In the majority of cases, our recommended approach outperformed LOF and DBSCAN techniques. After pre-processing with our technique, the accuracy is shown to rise and the KNN algorithm performs better overall. It maintains first place with a significant disparity.

Ultimately, our proposed approach can be used for both short and lengthy texts and can be applied to most datasets without taking into account the system knowledge. Ultimately, the versatility of our approach may be increased by making use of additional similarity methods, such semantic similarity.

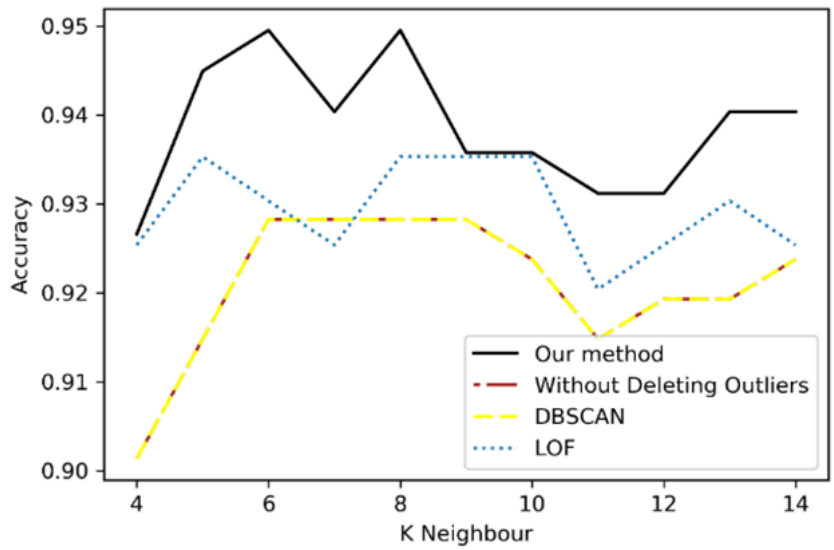


Figure 11: The accuracy of KNN on the BBC dataset with k between 4 to 14 with N-gram vectors

## References

[1] F. Abedini, M. Bahaghighat, M. S'hoyan, Wind turbine tower detection using feature descriptors and deep learning. *Facta Universitatis, Series: Electronics and Energetics*, **33,** 1 (2019) 133–153. ⇒105

[2] J. Allan, V. Lavrenko, D. Malin, R. Swan, Detections, bounds, and timelines: Umass and tdt-3. In *Proceedings of Topic Detection and Tracking Workshop*, pp. 167–174. Citeseer, 2000. ⇒92

[3] M. Bahaghighat, F. Abedini, Q: Xin, M. Mohammadi Zanjireh, S. Mirjalili, Using machine learning and computer vision to estimate the angular velocity of wind turbines in smart grids remotely. *Energy Reports*, **7** (2021) 8561–8576. ⇒92

[4] M. Bahaghighat, Q. Xin, S. Ahmad Motamedi, M. Mohammadi Zanjireh, A. Vacavant, Estimation of wind turbine angular velocity remotely found on video mining and convolutional neural network. *Applied Sciences*, **10,** 10 (2020) 3544. ⇒105

[5] C. Barreyre, L. Boussouf, B. Cabon, B. Laurent, J-M. Loubes, Statistical methods for outlier detection in space telemetries. *Space Operations: Inspiring Humankind's Future*, pp. 513–547, 2019. ⇒93

[6] I. Ben-Gal, *Outlier detection in: Data mining and knowledge discovery handbook: A complete guide for practitioners and researchers*, 2005. ⇒93

[7] Y. Bengio, O. Delalleau, C. Simard, Decision trees do not generalize to new variations. *Computational Intelligence*, **26,** 4 (2010) 449–467. ⇒100

[8] M. Bozorgi, M. Mohammadi Zanjireh, M. Bahaghighat, Q. Xin, A time-efficient and exploratory algorithm for the rectangle packing problem. *Intelligent Automation & Soft Computing*, **31,** 2 (2022) 885–898. ⇒92

[9] A. Z. Broder, S. C. Glassman, M. S Manasse, G. Zweig, Syntactic clustering of the web. *Computer networks and ISDN systems*, **29,** 8–13 (í997) 1157–1166. ⇒98

[10] M. Ester, H-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, vol. 96, pp. 226–231, 1996. ⇒93

[11] M. Ghorbani, M. Bahaghighat, Q. Xin, F. Özen, ConvLSTMconv network: a deep learning approach for sentiment analysis in cloud computing. *Journal of Cloud Computing*, **9,** Article no: 16 (2020). ⇒92, 105

[12] J. Guzman, B. Poblete, On-line relevant anomaly detection in the twitter stream: an efficient bursty keyword detection model. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pp. 31–39, 2013. ⇒92, 94

[13] A. Hajikarimi, M. Bahaghighat, Optimum outlier detection in internet of things industries using autoencoder. In *Frontiers in Nature-Inspired Industrial Optimization*, pp. 77–92, 2022. ⇒92

[14] D. J. Higham, An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, **43,** 3 (2001) 525–546. ⇒100

[15] T. K. Ho, Random decision forests. In *Proc. of 3rd Int. Conf. on Document Analysis and Recognition*, vol. 1. pp. 278–282. IEEE, 1995 ⇒99

[16] V. Hodge, J. Austin, A survey of outlier detection methodologies. *Artificial Intelligence Review*, **22** (2004) 85–126. ⇒92

[17] M. Jamalzadeh, M. Maadani, M. Mahdavi, Ec-mopso: an edge computing-assisted hybrid cluster and mopso-based routing protocol for the internet of vehicles. *Annals of Telecommunications*, **77,** 7–8 (2022) 491–503. ⇒93

[18] S. M. Jameii, M. Maadani, Intelligent dynamic connectivity control algorithm for cluster-based wireless sensor networks. In *2016 11th Int. Conf. for Internet Technology and Secured Transactions (ICITST)*, pp. 416–420. IEEE, 2016. ⇒93

[19] T. Joachims, *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization.* Technical Report, Carnegie-Mellon Univ. Pittsburgh. Dept. of Computer Science, 1996. ⇒98

[20] S. Kannan, V. Gurusamy, S. Vijayarani, J. Ilamathi, Ms. Nithya, S. Kannan, V. Gurusamy, Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, **5,** 1 (2014) 7–16. ⇒92

[21] F. Khorasani, M. Mohammadi Zanjireh, M. Bahaghighat, Q. Xin, A tradeoff between accuracy and speed for k-means seed determination. *Comput. Syst. Sci. Eng.*, **40,** 3 (2022) 1085–1098. ⇒92

[22] B. S. Kumar, V. Ravi, A survey of the applications of text mining in financial domain. *Knowledge-Based Systems*, **114** (2016) 128–147. ⇒92

[23] R. Kumaraswamy, A. Wazalwar, T. Khot, J. Shavlik, S. Natarajan, Anomaly detection in text: The value of domain knowledge. In *The Twenty-Eighth International Flairs Conference*, 2015. ⇒92

[24] Y. Li, Z. Chen, D. Zha, K. Zhou, H. Jin, H. Chen, X. Hu. Autood: Automated outlier detection via curiosity-guided search and self-imitation learning. *arXiv preprint arXiv:2006.11321*, 2020. ⇒92

[25] Y. Liu, Z. Li, Ch. Zhou, Y. Jiang, J. Sun, M. Wang, X. He, Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, **32,** 8 (2019) 1517–1528. ⇒93

[26] A. R. Lubis, M. Lubis, et al., Optimization of distance formula in k-nearest neighbor method. *Bulletin of Electrical Engineering and Informatics*, **9,** 1 (2020) 326–338. ⇒99

[27] H. P. Luhn, A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, **1,** 4 (1957) 309–317. ⇒98

[28] M. Norouzi Shad, M. Maadani, M. Nesari Moghadam, Gapso-Svm: an IDSS-based energy-aware clustering routing algorithm for IoT perception layer. *Wireless Personal Communications*, **216** (2022) 2249–2268. ⇒93

[29] M. Oghbaie, M. Mohammadi Zanjireh, Pairwise document similarity measure based on present term set. *Journal of Big Data*, **5,** 1 (2018) 1–23. ⇒98

[30] M. Platakis, D. Kotsakos, D. Gunopulos, Searching for events in the blogosphere. In *Proceedings of the 18th Int. Conf. on World Wide Web*, pp. 1225–1226, 2009. ⇒92

[31] X. Qin, L. Cao, E. A. Rundensteiner, S. Madden, Scalable kernel density estimation-based local outlier detection over large data streams. In *Proceedings of the 22nd Int. Conf. on Extending Database Technology (EDBT)*, 2019. ⇒93

[32] J. P. Reiter, T. E. Raghunathan, The multiple adaptations of multiple imputation. *Journal of the American Statistical Association*, **102,** 480 (2007) 1462–1471. ⇒99

[33] M. Rostami, M. Bahaghighat, M. Mohammadi Zanjireh, Bitcoin daily close price prediction using optimized grid search method. *Acta Universitatis Sapientiae, Informatica*, **13,** 2 (2021) 265–287. ⇒92

[34] S. N. Sajedi, M. Maadani, M. Nesari Moghadam, F-leach: a fuzzy-based data aggregation scheme for healthcare IoT systems. *The Journal of Supercomputing*, **78,** 1 (2022) 1030–1047. ⇒92

[35] E. Schubert, M. Weiler, H-P. Kriegel, Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. In *Proceedings of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 871–880, 2014. ⇒92

[36] H. Schütze, Ch. D. Manning, P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008. ⇒98

[37] A. Shamseen, M. Mohammadi Zanjireh, M. Bahaghighat, Q. Xin, Developing a parallel classifier for mining in big data sets. *IIUM Engineering Journal*, 22, 2 (2021) 119–134. ⇒92, 95

[38] M: Templ, J. Gussenbauer, P. Filzmoser, Evaluation of robust outlier detection methods for zero-inflated complex data. *Journal of Applied Statistics*, 47, 7 (2020) 1144–11673. ⇒92

[39] B. Wang, J. Sharma, J. Chen, P. Persaud, Ensemble machine learning assisted reservoir characterization using field production data–an offshore field case study. *Energies*, 14, 4 (2021) 1052. ⇒101

[40] Y. Wu, X. Li, F. Luan, Y. He, A novel gpr-based prediction model for strip crown in hot rolling by using the improved local outlier factor. *IEEE Access*, **9** (2020) 458–469. ⇒94

[41] Y. Yan, L. Cao, C. Kulhman, E. Rundensteiner, Distributed local outlier detection in big data. In *Proceedings of the 23rd ACM SIGKDD Int. Conference on knowledge Discovery and Data Mining*, pp. 1225–1234, 2017. ⇒92, 93

[42] Y. Zhao, Z. Nasrullah, Z. Li, PyOD: A Python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588*, 2019. ⇒92

*Received: May 5, 2023* • *Revised: June 21, 2023*