International Journal of Communication Networks and Information Security
ISSN: 2073-607X, 2076-0930
Volume 15 Issue 02 Year 2023

IoT-Based Data Size Minimization Using Cluster-Based-Similarity-Elimination

Alaa A. Abdelhafez*
Teaching Assistant,Faculty of Computers and Artificial Intelligence, Giza, Egypt
a.abdelhafez@fci-cu.edu.eg
Osama Ismael
Doctor,Faculty of Computers and Artificial Intelligence, Giza, Egypt
osama@fci-cu.edu.eg
Hatem Elkady
Professor, Faculty of Computers and Artificial Intelligence, Giza, Egypt
hkadi@fci-cu.edu.eg

Article History

Received: 14 August 2023
Revised: 10 September 2023
Accepted: 19 October 2023

Abstract

This paper proposes a new redundancy reduction approach for continuous data flow from IoT devices based on minimizing the size of IoT data using a novel cluster-based-similarity-elimination algorithm. The continuously flowing data from IoT devices are characterized by the existence of redundant records. This redundancy not only leads to the overfitting of models but also requires a large processing power because of the large number of records. Feature selection is a technique used to partially reduce the data and thus redundancy, however, this is not sufficient. Removing redundant data is considered of utmost importance because as smart city scenarios are implemented, flow data generation requires more advanced analytics to deal with the evolution and regrowth of the IoT environment. Thus, this study aims to minimize processing time while maintaining the best accuracy by minimizing data similarity, therefore addressing the overfitting problem, and saving time. The proposed approach minimizes the data size, considering the number of tuples. The effectiveness of the proposed approach was validated using various classification algorithms and evaluation metrics. The results show a significant improvement compared with traditional approaches, resulting in a reduction in the real-time classification execution time to only 9% of the original time. This approach can be used to optimize data size and achieve accurate results with a fast execution time while also addressing overfitting issues.

Keywords: Data Minimization, IoT Data Analytics, Data Real-Time Analytics, Machine Learning (ML), Classification Task, Data Optimization.

1. Introduction

The Internet of Things integrates an enormous number of smart devices and uses the data generated to make informed decisions. However, raw data from these devices is often of little value and requires specific analytical methods to extract meaningful insights. For this purpose, two methods are used [1]: real-time and offline analytics.

Rapid real-time data analytics approaches are crucial when processing streamed sensor data, and many researchers have investigated machine learning algorithms to solve this problem [2], [3], [4], [5], [6], particularly in the area of classification. To improve the time required to achieve real-time classification with high accuracy, the methodology focuses on the classification task by creating a pre-classification phase designed to reduce the data size.

Regarding the Internet of Things (IoT), the sheer velocity of data production poses an even greater challenge. The performance of any system is directly influenced by the number of features that may cause many duplicates and the volume of input records. In addition, most IoT devices are resource-constrained, making data processing and analysis even more difficult. In the context of IoT data, deduplication enhances data features by identifying and eliminating exact duplicates, thus improving accuracy. In contrast, similarity reduction focuses on refining data records by detecting patterns and relationships, leading to benefits such as similarity reduction that contributes to significant time and storage savings. Both aspects offer distinct advantages: deduplication ensures feature accuracy, whereas similarity reduction provides valuable insights and efficiency gains. Challenges encompass computational complexity and data variability for deduplication, and interpretation complexity and scalability for similarity reduction. Achieving a harmonious balance between these aspects is pivotal for optimizing the full potential of IoT data.

To overcome these challenges, researchers have turned to the field of data reduction to minimize the data size. Some researchers have focused on modifying classification algorithms [7] to improve prediction accuracy, whereas others have focused on feature selection [8] to reduce the time required to complete the task. In this study, the proposed method focuses on optimizing the number of records by eliminating duplicates. Duplicate data increases processing time and can cause overfitting problems. The time required to process all the readings is affected by network latency between the sensors and applications.

Solving the overfitting problem is of utmost importance because it leads to inaccurate predictions and decisions.

There is redundancy in the weather data, with similar values for various attributes. The dataset can be optimized using the proposed cluster-based similarity elimination algorithm to remove duplicate records and reduce the data size without losing any valuable information, thus avoiding the overfitting problem. By reducing overfitting, a more robust and reliable model can be created that can accurately predict the weather in Giza based on the given features.

As mentioned earlier, the proposed algorithm optimizes the data size before applying the classification algorithm by minimizing the redundancy in the data. The cluster-based similarity elimination algorithm is used to minimize the data size by removing duplicate records, in addition to the usual minimization of the number of features.

Several experiments were performed using different ML algorithms to evaluate the performance of the proposed approach and compare it with traditional approaches and related work techniques.

2. Related Work

Several studies have explored different approaches for feature selection [8], such as wrapper-based methods, correlation-based methods, and evolutionary algorithms. Additionally, in [2], [3], [4], [5], and [6], the authors performed comparisons between different machine learning algorithms, including both traditional and deep learning models, for a range of applications such as activity recognition, IoT data classification, and diabetes prediction. Table 1 summarizes the approaches and classification algorithms that were adopted and the results, relevant literature, identifying the strengths and limitations of each:

Table 1. Summary Of Relevant Classification Algorithms In Literature

Algorithm	Results
DT (Decision Tree)	[5], DT achieved an accuracy of 76.70% on the Pima Indian diabetes dataset. [3], DT achieved an accuracy of 80.9%. [6], DT accuracy on the Pima Indian diabetes dataset is 76.61%. [4], DT accuracies reported in the paper in offline mode and real-time mode are Offline mode: 92.76%, Real-time mode: 83.24%.
Logistic Regression (LR)	[2], LR Achieved an accuracy of 95.1%.
ANN (Artificial Neural Network)	[10], ANN Achieved an accuracy of 77.34%. [2], ANN Achieved an accuracy of 98.1%. [3], ANN achieved an accuracy of 83.8%. [5], ANN achieved an accuracy of 78% on the Pima Indian diabetes dataset. [4], ANN accuracies reported in the paper in offline mode and real-time mode are Offline mode: 94.11%, Real-time mode: 78.32%.
RF (Random Forest)	[6], RF accuracy on the Pima Indian diabetes dataset is 79.39%. [5], RF achieved the best performance with an accuracy of 79.10%. [2], RF Achieved an accuracy of 97.4%. [4], RF accuracies reported in the paper achieved the highest accuracy in offline mode, and real-time mode are Offline mode: 97.23%, Real-time mode: 82.34%.
KNN (K-Nearest Neighbors)	[3], KNN achieved an accuracy of 81.6%. [5], KNN achieved an Accuracy of 76.30% on the Pima Indian diabetes dataset. [6], KNN accuracy on the Pima Indian diabetes dataset is 76.24%. [4], KNN accuracies reported in the paper in offline mode and real-time mode are Offline mode: 91.32%, Real-time mode: 81.23%.
GNB (Gaussian Naïve Bayesian)	[9], GNB achieves an average accuracy of 76.9%, Across 20 datasets. [3], GNB achieved an accuracy of 80.2%. [6], GNB accuracy on the Pima Indian diabetes dataset is 75.88%.

New methods for improving the performance of existing machine learning algorithms have also been proposed [9], [7], [10]. [9] showed that the effectiveness of the machine learning algorithm depends on the nature of the data. Using the Evo-Bagging, it was shown in [7] to have superior performance compared with other ensemble learning methods. Deep Learning (DL) achieves better performance than traditional machine learning methods [10].

All of [2], [3], [4], [5], [6] compared different machine learning and deep learning algorithms for various applications. The papers investigated the performance of different algorithms by analyzing their accuracy, precision, recall, and F1-score. The algorithms were applied to different datasets for tasks such as activity recognition and diabetes prediction. They found that deep learning algorithms outperformed traditional machine learning algorithms in terms of accuracy. One drawback of these studies is that they are specific to certain datasets and applications and may not be generalized to other datasets or applications. Another drawback is that the performance of the algorithms is heavily dependent on the quality of the data used for training and testing.

To summarize, the related works mentioned above have focused on improving classification tasks through feature selection. However, only a few studies have investigated IoT or real-time data, and none have attempted to minimize the data size vertically based on the number of records. The model proposed in this study aims to minimize execution time while producing accurate results by horizontally reducing data size through feature selection and vertically reducing the number of records.

The primary focus is on the classification tasks essential for various IoT applications. This work proposes a novel hybrid technique called "Cluster-Based-Similarity Elimination algorithm (CBSE)". The proposed approach is compared with different ML algorithms using different evaluation metrics.

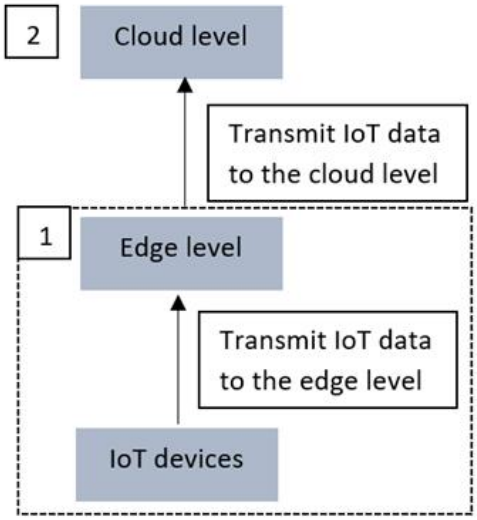

Figure 1. The Whole Image

3. Methodology

The high speed at which IoT data are generated poses difficulties in terms of processing and analysis. Minimization becomes crucial for real-time analytics of IoT data. While some researchers concentrate on reducing data size through feature selection or extraction, none have yet explored the simultaneous reduction of both feature and record numbers. This is precisely the objective of the proposed methodology: to achieve comparable results in a shorter period.

In this section, a detailed demonstration of the proposed technique is presented, which is part 2 of Figure 1. The proposed methodology focuses on data obtained from IoT devices or edges, as demonstrated in the previous section. All data processing is performed in the cloud, specifically using Google CoLab. The resulting model can be loaded onto the edge device or directly transmitted to it. The proposed technique comprises two sub-steps aimed at achieving valuable feature selection and minimizing the number of tuples. Initially, the most relevant features are selected, followed by cluster-based-similarity elimination algorithm to identify the minimum number of tuples required. The proposed approach is specifically designed for real-time IoT data analytics, offering high efficiency and accurate results within a short timeframe. Moreover, the algorithm to select tuples that contain significant information is utilized, facilitating real-time classification and prediction using IoT streaming data through the cluster-based similarity elimination algorithm.

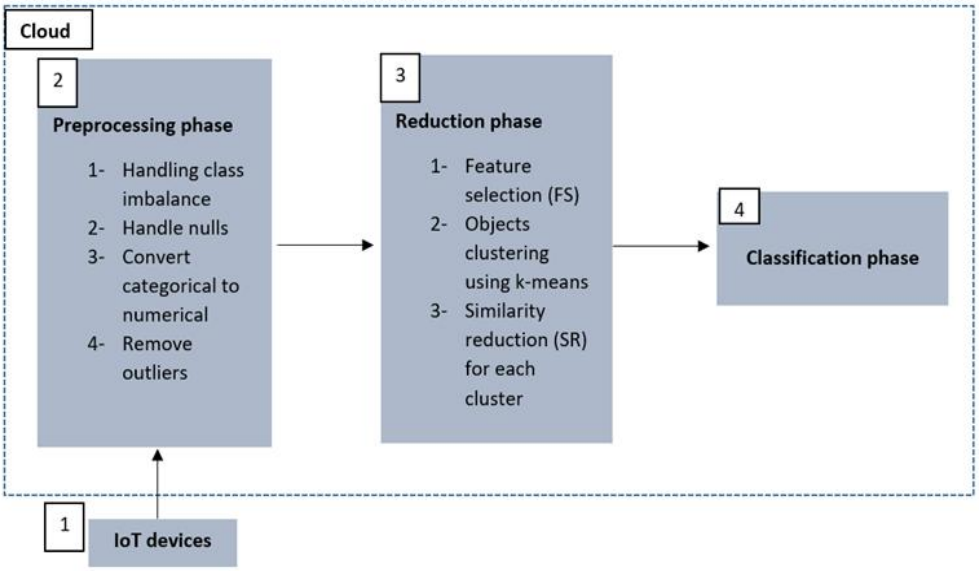

Figure 2. Classification Based on Cluster-Based-Similarity Elimination Algorithm

To address the aforementioned issues, a hybrid technique called CBSE is introduced as mentioned above. This approach encompasses three stages: pre-processing, minimization, and classification, as depicted in Figure 2 (step #1 is the raw data acquisition step, and it is out of scope and will not be detailed in this paper). In the initial stage, the data undergoes preprocessing to ensure its suitability for modeling (step #2). This involves four crucial procedures: handling class imbalance, treating null values, converting categorical data into numerical format, and removing outliers. These steps play a pivotal role in enhancing result quality. In the second stage, step 3, the proposed methodology, the CBSE, is used which is a compound of k-means clustering and Similarity Reduction (SR), to select the most effective features and records that represent the data, achieving precise outcomes in the shortest possible modeling time. The data are minimized and condensed in both dimensions: width (number of features) and length (number of records). This approach makes the data more manageable. Finally, in the third stage (step 4), the classification algorithm is employed. Each phase on the cloud is described in detail in the following subsections.

3.1 Preprocessing Phase

To ensure the accuracy and quality of the results, the preprocessing phase is a crucial step in the proposed methodology. This phase is divided into four steps, which are illustrated in Fig 2: handling class imbalance, handling nulls, converting categorical to numerical, and removing outliers. The first step in the preprocessing phase is handling class imbalance, where we address the issue of imbalanced datasets using the oversampling technique (using the sklearn.utils library). The second step, nulls, is handled depending on the specific features. The third step involves converting categorical fields to numerical ones using the LabelEncoder library from the Python scikit-learn. The fourth and the last step, outliers were detected and removed using the InterQuartile Range (IQR) method.

3.2 Cluster-based-similarity-elimination Algorithm

Once the data preprocessing is completed, move on to the next stage, where the methodology, the CBSE comes into play. Our aim is to identify the most suitable records that accurately reflect the data while achieving faster results. Phase two consists of two stages: Feature Selection (FS) and the CBSE technique, as demonstrated in Fig 2. These two stages work together to optimize and streamline the data size in both dimensions, width, and length.

To minimize the time required to find similar records, first stage; RF feature selection is used [8], which is an effective method to avoid overfitting. The choice of feature selection method is critical as one feature may have considerable redundancy and can cause overfitting problems, thus, affecting the CBSE performance later.

The Second stage, cluster-based-similarity elimination algorithm, reduces the number of similarity matches. Initially, cluster the data using k-means with different cluster sizes and select the best clustering number based on the scores of the k-means curve. Then, choose the number of clusters

on the elbow that will be the best number for this dataset. Employing the Gaussian distribution—an elegantly symmetric probability distribution centred around the mean, emphasizing a higher frequency of occurrences near the mean than at greater distances—we calculate deviations. This involves the use of 2 times the standard deviation (SD) to encompass approximately 95% of the dataset within the mean range. Our application of this principle is conducted across distinct clusters, where we gage deviations between the centroid and 2*SD. Having organized the dataset into these clusters, we embarked on a cluster-by-cluster iteration. Within each cluster, we assessed the correlation between individual pairs of records, assuring that none of these points exceeded the specified deviation limit. This strategic approach optimizes the calculation of record similarities, focusing on the region in which the Gaussian distribution certifies the concentration of 95% of cluster data. By doing so, we not only enhance efficiency but also harness the intrinsic concentration guarantee of the Gaussian distribution. The correlation is calculated using the Pearson correlation statistical measure, removing records that are too similar or have similarity up to 90%. This process avoids comparing different records and saves time while processing CBSE, which uses the following equation:

$$R = \frac{\sum (X-\bar{x})(Y-\bar{y})}{\sqrt{\sum (X-\bar{x})^2 . \sum (Y-\bar{y})^2}}$$

Where, $\bar{x}$ = mean of X variable, $\bar{y}$ = mean of Y variable

The value of Pearson's correlation coefficient ranges from -1 to 1. The correlation coefficient can be interpreted as follows:
Correlation = -1 indicates a high negative association,
Correlation = 0 indicates no association,
Correlation = 1 is a very strong positive relationship.
If they are similar or have similarity up to 90%, one of them is removed and keep the compared record, and so on to compare all the records as pairs.

The Pearson correlation statistical measure assumes that each variable should be continuous and that outliers should be absent. Thus, outliers are removed from the data before applying this phase.

3.3 Classification Phase

The last phase is the classification phase, which is critical for various IoT applications. Data classification is generally characterized as the process of organizing data by relevant categories so that it can be used more effectively in prediction. The proposed technique is examined with 6 different classification algorithms used in the related work, which we compared with Logistic Regression (LR) [11], Decision Trees(DT) [12], Random Forests(RF) [13], K-Nearest Neighbors(KNN) [14], Gaussian Naïve Bayesian(GNB) [9], and Artificial Neural Network(ANN) [9].

Logistic Regression is a fundamental statistical method for binary classification, widely used in machine learning applications such as healthcare and finance. It models the probability of a binary outcome using the sigmoid function, transforming a linear combination of input features and weights into a probability score between 0 and 1. Training involves iteratively adjusting the weights and bias to minimize a loss function, often cross-entropy, using techniques such as gradient descent.

A Decision Tree is a prominent machine learning algorithm used for both classification and regression tasks. It operates by recursively partitioning the input data into subsets based on the most discriminative features, creating a tree-like structure of decision and leaf nodes. Each internal decision node represents a feature and a corresponding decision rule, whereas each leaf node represents a predicted outcome. Decision Trees are highly interpretable and can handle both numerical and categorical data. They are constructed through a process that selects features to split the data based on metrics such as Gini impurity or information gain, aiming to maximize the homogeneity of the resulting subsets. This process is repeated recursively until a stopping criterion is met, often leading to easily understandable rules for prediction. While Decision Trees can overfit, techniques such as pruning, and ensemble methods (Random Forests, Gradient Boosting) are employed to enhance their performance and generalization.

Random Forest is a powerful ensemble learning technique in machine learning, widely used for classification, regression, and anomaly detection tasks. It is built upon the foundation of Decision Trees. A Random Forest algorithm creates multiple Decision Trees, each trained on a different subset of data and using a subset of features. These trees operate independently and make predictions. The

final prediction from the Random Forest is obtained through a majority vote (for classification) or averaging (for regression) of the predictions made by individual trees. This ensemble approach mitigates overfitting and enhances the model's generalization ability, as the combined decisions of multiple trees tend to be more robust and accurate than a single tree's decision. Random Forest also provides a measure of feature importance, helping to identify the features that are influential in making predictions.

K-Nearest Neighbors (KNN) is a simple, yet effective supervised machine learning algorithm used for classification and regression tasks. KNN operates on the principle of proximity: it classifies or predicts a data point based on the majority class or average value of its K nearest neighbors in the feature space. The algorithm does not involve explicit model training; instead, it memorizes the entire training dataset. When making predictions for new data points, the KNN calculates distances (often using Euclidean distance) between the input features and training data points. It then selects the K nearest neighbors and determines the prediction based on their classes (for classification) or their average value (for regression). K, the number of neighbors, is a hyperparameter that impacts the algorithm's performance and can be chosen through techniques such as cross-validation. While KNN is intuitive and easy to understand, it can be sensitive to the choice of K and the scale of features.

Gaussian Naive Bayes is a probabilistic machine learning algorithm often used for classification tasks, particularly when dealing with continuous-valued features. It is based on Bayes' theorem and assumes that the features are conditionally independent given the class label. Despite its "naive" assumption of feature independence, Gaussian Naive Bayes can perform remarkably well in practice and is computationally efficient. The algorithm models the likelihood of each feature's values within each class as a Gaussian (normal) distribution and calculates the posterior probability of each class given the observed feature values. During prediction, the class with the highest posterior probability is chosen. Gaussian Naive Bayes is especially suitable for text classification and sentiment analysis tasks. Although it might not capture complex relationships in data, it serves as a quick and interpretable classification method.

Artificial Neural Networks (ANNs) are versatile machine learning models inspired by the structure of biological neurons, widely employed for tasks such as classification, regression, image recognition, and natural language processing. ANNs consist of interconnected layers—input, hidden, and output—where weights determine the strength of connections. During training, these weights are adjusted via techniques such as gradient descent, driven by chosen loss functions. Activation functions introduce nonlinearity which is crucial for capturing complex patterns. Deep Learning, a subset of ANNs, involves models with multiple hidden layers with Excel in tasks such as image analysis. Convolutional Neural Networks (CNNs) specialize in image tasks by learning features through convolutional layers. Recurrent Neural Networks (RNNs) manage sequential data by retaining context. Whereas ANNs achieve remarkable results, they require substantial data and computation.

In addition, multiple evaluation metrics are used such as Accuracy, Precision, Recall, F1-score, and Time to evaluate the results, and high results for different classifiers were achieved.

By combining cluster-based-similarity elimination algorithms, advanced classification techniques, and real-time analytics, are all used to achieve superior accuracy, speed, and efficiency in handling IoT data. This innovative approach promises to revolutionize the field, enabling efficient and precise decision-making based on real-time insights from IoT streaming data. The experimental results in the upcoming section are presented.

4. Results and Discussion

In this section, we assess the proposed approach using the weatherAUS dataset [15]; the evaluation metrics used are accuracy, recall, f1-score, and precision. In addition, the execution time represents an important evaluation criterion.

All the experiments were conducted on a system with an Intel Corei7 CPU at 2.9GHz, 16 GB of RAM, 512 SSD of secondary storage, running Windows 10 and Python 3.7. Google Colab uses IDE for model development and leverages machine and deep learning libraries and packages in scikit-learn.

4.1 Data Set

WeatherAUS dataset is the dataset we used, it contains approximately 10 years of daily weather observations from many locations across Australia. It has twenty-three features and 145460 rows.

Observations of a huge amount of atmosphere at Australia's stations. It is a huge dataset of values collected from numerous weather sensors and equipment metering. The collected data are about temperature, rainfall, evaporation, sunshine, direction, and speed of wind, humidity, pressure, and clouds. Most of the features are read at 9 a.m. and 3 p.m. This dataset is for a binary classification problem, which can be used to predict if it is rainy tomorrow or not. The smart environments created using IoT-compliant environment devices and supplies are the owners of the weatherAUS dataset. Rain Tomorrow is the target variable to predict. It means - did it rain the next day, Yes or No? This dataset required more preprocessing tasks; because it has 23 features, 3 of which are categorical with 16 different values. Below are descriptive visualizations to make the dataset more understandable.

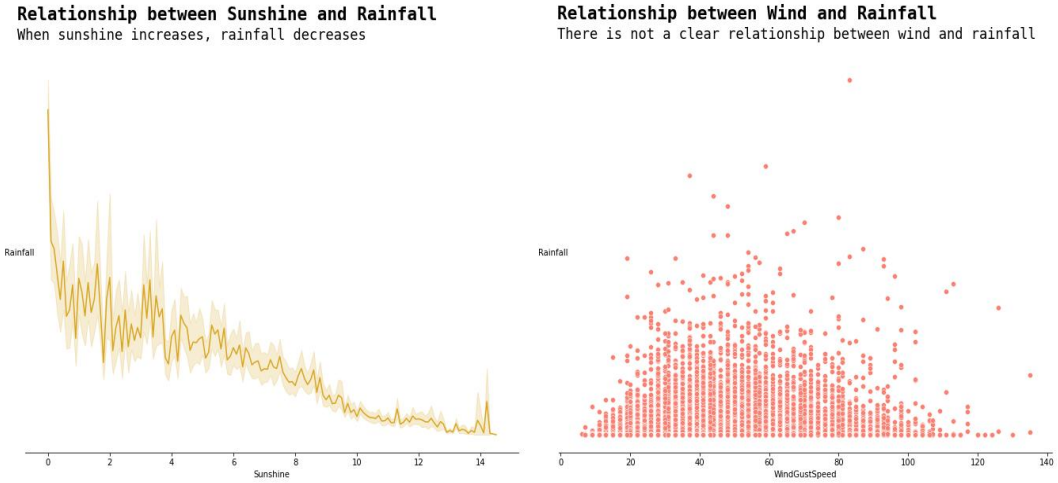

Figure 3. The Relationship Between the Sunshine, Rainfall and the Wind, Rainfall in WeatherAUS Dataset

Figure 3 shows that sunshine and rainfall are inherently interconnected, as they play pivotal roles in shaping the climate. Typically, an increase in sunshine often correlates with lower levels of rainfall, signifying drier and sunnier conditions. Conversely, a decrease in sunshine is frequently associated with higher rainfall, indicating wetter and overcast weather patterns. In addition, the WeatherAUS dataset provides a comprehensive perspective on the intricate relationship between wind and rainfall in Australia's ever-changing weather patterns. Wind is a fundamental factor that influences rainfall in several ways. Prevailing wind patterns can transport moist air masses, contributing to increased rainfall in specific regions. For instance, onshore winds from the ocean can bring moisture-laden air to coastal areas, promoting rainfall. In contrast, offshore winds may inhibit rainfall by pushing dry air masses into an area. Wind speed and direction also play a crucial role in the distribution and intensity of rainfall. Intense winds can enhance evaporation rates and disperse raindrops, potentially reducing overall precipitation in some areas.

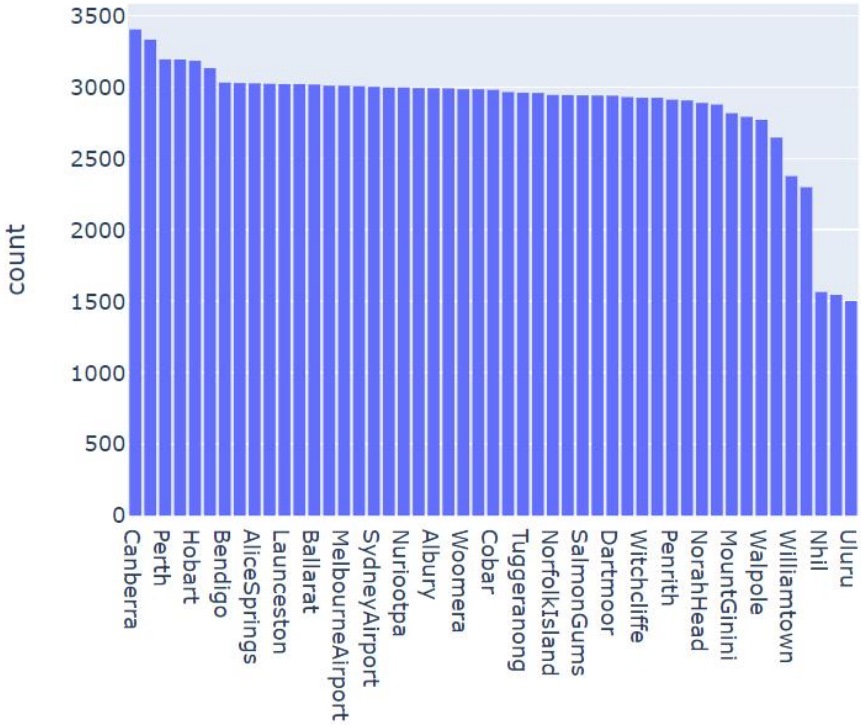

Figure 4. Count of Number of Times it Rained in Each City in WeatherAUS Dataset

In the Figure 4, the count of the number of times rain occurred in each city provides essential insights into the precipitation patterns across different regions of Australia. Some cities may experience frequent rainfall throughout the year, whereas others may experience more sporadic or seasonal precipitation. Understanding these local variations in rain occurrence helps in assessing vulnerability to droughts, floods, and other weather-related challenges in specific areas, aiding in the development of region-specific strategies for water resource management and disaster preparedness.

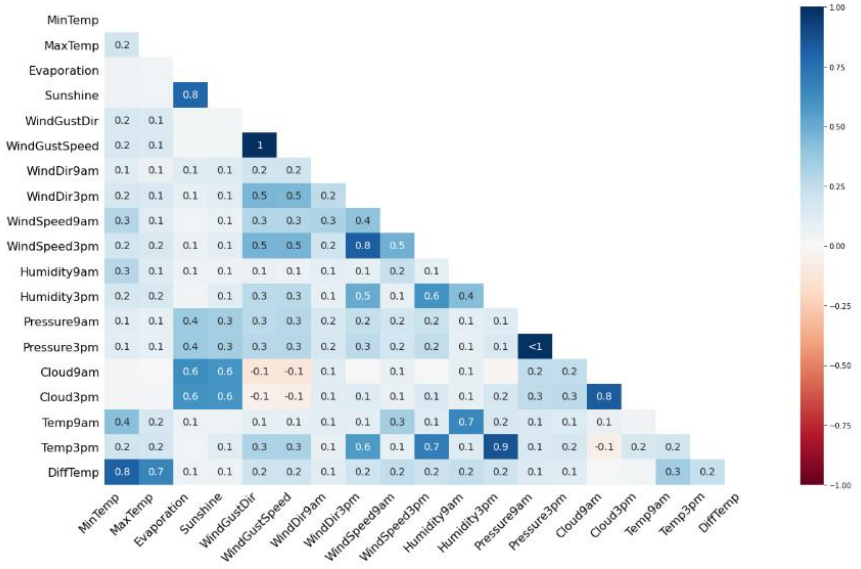

Figure 5. The Correlation Between Most of the Features in WeatherAUS Dataset

The dataset offers a rich opportunity to explore the correlation between various features, shedding light on the complex interplay of weather factors in Australia. In the Figure 5, these correlations can be analyzed to identify patterns and dependencies among variables such as temperature, humidity, wind speed, air pressure, and rainfall. Understanding these correlations is essential for developing predictive models, as it enables us to anticipate how changes in one weather variable might affect others, improving our ability to select the most key features.

4.2 Evaluation Metrics

Once machine learning has been applied, some methods are needed to assess its effectiveness. The performance metrics are the names of these tools. Studies have presented many metrics, each of which considers certain aspects of algorithm execution. Therefore, we require a suitable set of measurements for the performance evaluation of each machine learning task. In this study, several widely used metrics are employed for classification problems to obtain insightful information on algorithm performance and conduct a side-by-side analysis. These measures include the execution time in seconds, f1-score, accuracy, precision, and recall.

Deep models achieved high accuracy for some models but at the expense of longer execution times. The execution time to predict the class is of vital importance in the evaluation of IoT. As a result, we are not permitted to create models for IoT systems and devices that are extremely complicated and require many time-consuming computations. However, we must not overlook the fact that most IoT devices face resource and energy limitations.

4.3 Performance Comparison Using Evaluation Metrics

We planned to conduct many experiments to assess the effect of each phase of the proposed methodology on the performance of the classification algorithms. First, the experiments supported by the traditional method were executed (i.e., preprocessing + FS phases). Secondly, the experiments supported by the whole proposed methodology (i.e., the preprocessing phase and the complete phase 'CBSE'). Finally, we compared the proposed methodology with one of the related works to emphasize the time and storage saved.

4.4 Experiments With Traditional Method

In this section, different machine learning algorithms are applied to the given dataset after the preprocessing phase and FS are applied. Using the set of performance measures, Table 2 illustrates the experimental results of applying these machine learning algorithms.

Based on the performance analysis of the proposed algorithm, we discovered some intriguing findings. Among the algorithms tested, RF was the most accurate option, delivering exceptional results within a reasonable timeframe. On the other hand, GNB and DT exhibit impressive speed but slightly compromise accuracy, achieving success rates of 75%, and 86% respectively.

ANN, while demonstrating commendable accuracy, falls short of being the best owing to its resource-intensive nature. Despite the efforts of algorithm to optimize its performance, ANN still required approximately 367 s to process the vast amount of data, making it the second slowest algorithm after KNN.

Meanwhile, LR shows respectable speed and performs well in terms of accuracy, although it falls behind RF by a margin of 12%.

KNN also produces satisfactory accuracy, but its execution times surpass those of the other algorithms, making it less desirable in this context.

The competition for accuracy primarily revolves around the RF, ANN, and DT. These three algorithms showed comparable levels of accuracy, warranting further consideration. However, GNB, DT, and LR were the best in terms of time.

To ensure a comprehensive evaluation, the data are divided into a 75% training set and a 25% testing set.

Table 2. A Comparison of the Performance of Different Machine Learning Algorithms

Algorithm	Accuracy	Precision	Recall	F1-score	Time (in seconds)
LR	0.795	0.793	0.789	0.791	2.739
DT	0.860	0.858	0.861	0.859	0.522
ANN	0.887	0.886	0.885	0.886	367.342
RF	0.927	0.925	0.928	0.926	30.919
KNN	0.821	0.818	0.819	0.819	115.311
GNB	0.752	0.749	0.750	0.749	0.113

4.5 Choosing the Number of Clusters and Similarity Percentage to Apply Full CBSE

In this part of the experiments, we applied machine learning algorithms to support the entire proposed methodology (i.e., the preprocessing and the complete CBSE phases). To apply the k-means algorithm, the number of clusters required for the k-means algorithm were selected using the k-means curve method. Specifically, we used the scores of the k-means curve method. As shown in Figure 6, the elbow was achieved in four clusters.

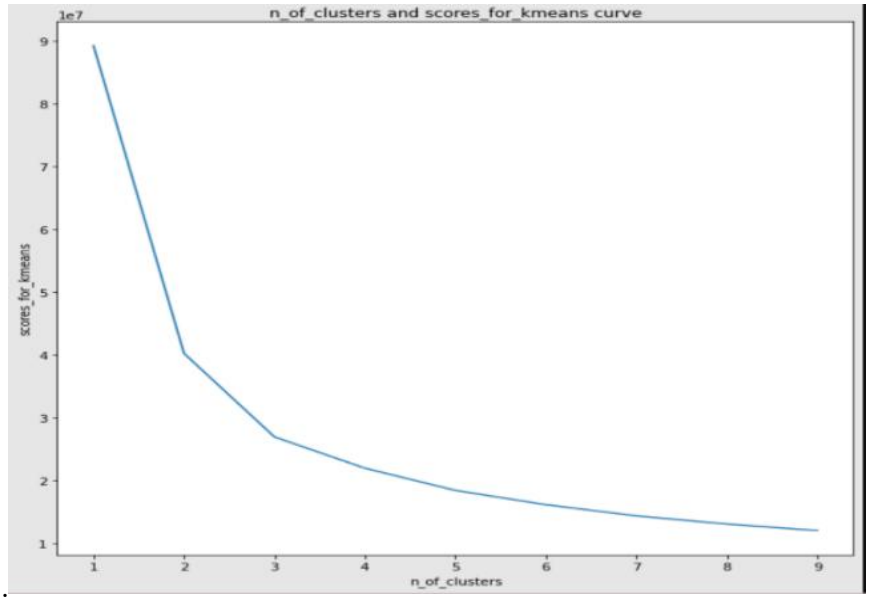

Figure 6. The Scores of k-means Curve Show the Best Number of Clusters for the WeatherAUS Data

In addition, the ML algorithms are assessed with the full algorithm with different similarity percentages equal to 60%, 70%, 80%, 90%, and 100% of the Pearson measure, and without the CBSE part, and the reduction in the size of the dataset is assessed. The results are shown in Figure 7, which shows a significant difference in the data size with and without the CBSE, and the same data size in all similarity measures; therefore, it is preferable to use 90 percent in similarity with this dataset.

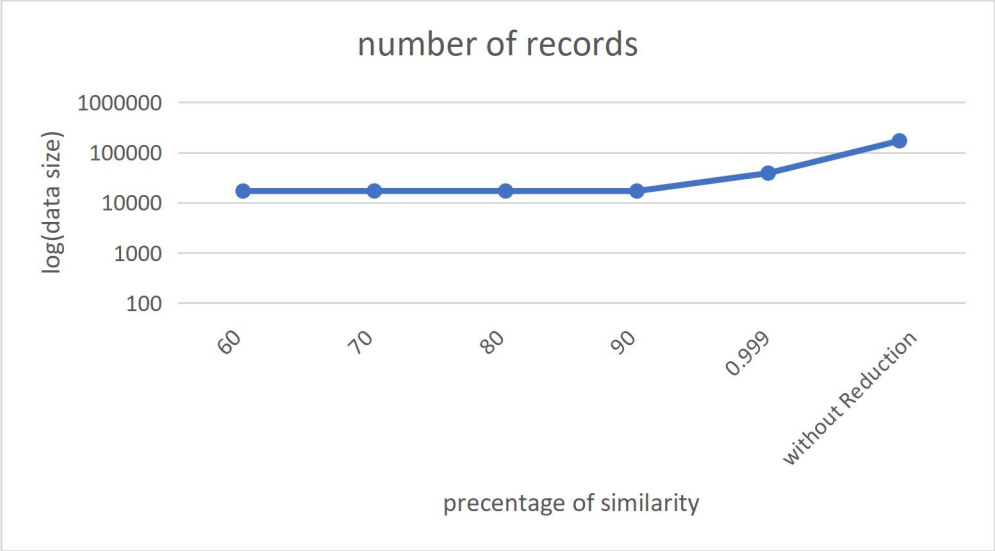

Figure 7. The Data Size with the Similarity Percentage and Without the CBSE

Table 3. A Comparison of the Performance of Different Machine Learning Algorithms with the Proposed Methodology using the Evaluation Equation

Algorithm	Accuracy	Precision	Recall	F1-score	Time (in seconds)
LR	0.920 (+0.160)	0.840 (+0.060)	0.600 (-0.230)	0.650 (-0.180)	0.080 (-0.970)
DT	0.910 (+0.060)	0.720 (-0.160)	0.740 (-0.140)	0.730 (-0.150)	0.030 (-0.930)
ANN	0.930 (+0.040)	0.790 (-0.110)	0.720 (-0.180)	0.750 (-0.150)	40.400 (-0.890)
RF	0.950 (+0.030)	0.920 (0.000)	0.770 (-0.170)	0.820 (-0.110)	1.800 (-0.940)
KNN	0.920 (+0.130)	0.820 (+0.010)	0.630 (-0.230)	0.680 (-0.170)	0.260 (-1.000)
GNB	0.850 (+0.140)	0.640 (-0.140)	0.730 (-0.030)	0.670 (-0.110)	0.010 (-0.890)

An evaluation equation is used to compare the traditional metrics to the CBSE, the equation= [(the Proposed Method accuracy – the Traditional Method accuracy)/the Traditional Method accuracy], this measure declares the percent of improvement in the measures, +1 indicates that this measure increases by +1 percent, -1 indicates that this measure decreases by -1 percent, for the time the negative means that the time decreased in seconds. As shown in Table 3, for the accuracy measure, all accuracy measures improved and increased. In addition, all the times decreased with a good improvement. However, some algorithms were improved, and some were not. The reason for this is that the size of the data available decreased and the number of positive classes to be predicted decreased. This is why this measure decreased compared with the traditional method with the full size of the data. The same reason for recall will be more affected by the size of the positive classes, sure for the f1-score too, because the f1-score is the average of both measures. This is a tradeoff between precision-recall and accuracy-recall, and we can see that to improve the accuracy, we should let some tradeoff with the recall measure. Like the precision, it can be seen that some algorithms are affected by the size minimization, and some are not; thus, the precision is good with some algorithms. It depends on the domain we work on, if we work on extremely sensitive data, and we care more about recall and precision measures with respect to accuracy and time improvement. We can choose the RF algorithm; this algorithm has the least effect on the minimization, no decrease mentioned, and the accuracy and time improved simultaneously.

According to the results shown in Table 3, the RF algorithm achieved the best performance across the different metrics, among its competitors, which is the only model that achieved 95% accuracy. Regarding the execution time, the lowest and highest model was 0.01 and 40.4 second, which belong to Gaussian Naïve Bayesian (GNB) and Artificial Neural Network (ANN), respectively. The RF algorithm achieved a moderate execution time, of 1.8 seconds. It is also noticeable that the

LR and ANN algorithms performed very well and achieved performance results very close to those of the RF algorithm. In addition, the DT algorithm achieved good execution time with acceptable performance measures.

Overall, all accuracies and execution times were enhanced after applying the CBSE algorithm. This is because CBSE picked only unique records, causing a very effective reduction in the data size. This is revealed in the enhanced performance of the classification algorithm, as well as the modeling was performed on 26% of the features, and approximately 10% of the records could be eliminated compared to the data size after the preprocessing phase of the rows, which is why we can see a significant improvement in the time.

As shown, the time with full CBSE is an average of 6% of the traditional time which is a very good improvement.

4.6 Time, Storage Saving and Comparison Results

A comparison between the performance of the proposed approach (CBSE) and the research by Vakili [11],

This study contributes to the field of machine learning and IoT by providing a comprehensive analysis and comparison of various machine and deep learning algorithms for IoT data classification.

This paper presents an experimental study that evaluates the performance of several machine and deep learning algorithms, including decision tree, k-nearest neighbors (KNN), random forest, support vector machine (SVM), multilayer perceptron (MLP), convolutional neural network (CNN), and long short-term memory (LSTM), on the WeatherAUS dataset. The study also provides insights into the impact of various parameters on the performance of the algorithms and identifies the best hyperparameters for each algorithm.

Overall, the contribution of this study lies in its comprehensive analysis and comparison of various machine and deep learning algorithms for IoT data classification, thereby providing valuable insights into the selection and optimization of algorithms for IoT data analysis and classification tasks. They used the same Weather dataset.

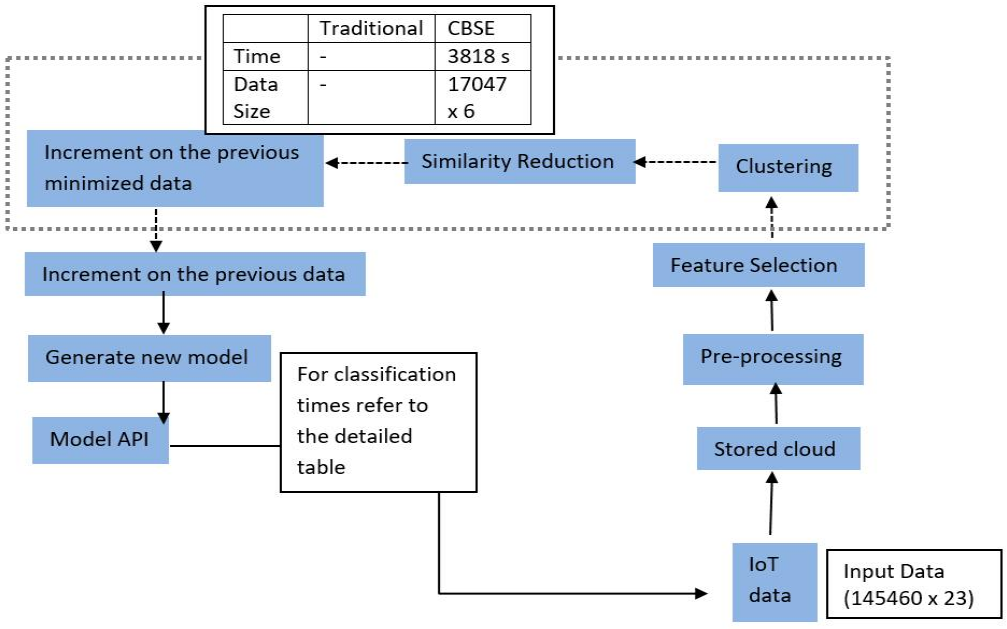

Figure 8. The Methodology Path Compared to [3]

Table 4. The Methodology Path Compared to [3]

	[3]	**CBSE for IoT Data**
Pre-processing	✓	✓
Feature Selection	✓	✓
CBSE	-	✓
Model generation	✓	✓
Classification	✓	✓

Table 4 presents a comparison between related studies [3] and CBSE for IoT data. In terms of preprocessing and feature selection, both methodologies, represented by checkmarks (✓), employ these steps. However, when it comes to clustering-based-similarity elimination, [3] does not incorporate these techniques, whereas CBSE utilizes them, as indicated by a checkmark (✓). Both methodologies demonstrate model generation and classification as part of their processes, with checkmarks (✓) indicating inclusion. This table highlights the differences between the two approaches, emphasizing the advantages of CBSE for IoT data in terms of incorporating clustering and similarity-reduction techniques. Figure 8 is a graphical presentation for both methodologies, first the input data come from the IoT devices and loaded to the cloud all steps applied for the traditional way, the time, and data size the same after preprocessing, as well as, after FS. The difference here in the dashed box, the traditional way goes to the "increment on the previous data step" and generates the new model, but CBSE goes to "increment on the previous minimized data" and generates the new model.

First run:

	Traditional	**CBSE**
Data size	170669 x 6 (history1)	17047 x 6 (history2)

In the initial run, the traditional methodology employed the entire data size during model generation, resulting in a longer time required to build the model (Table 5). Conversely, when utilizing the CBSE approach, the data size is reduced to only 9.99% of the traditional size. In subsequent runs, when updating the model with new data from sensors, this history (referred to as history1 and history2) will be utilized to increase the data. This approach allows for more efficient and streamlined model updates, leveraging the existing data history to incorporate new information.

Second run:

	Traditional	CBSE
Data size	history1 + around (170669 x 6)	history2 + around (17047 x 6)

In the second run, an intriguing transformation unfolds as the data undergo a process of size optimization and minimization, with long-term implications for subsequent data updates. It is important to note that this reduction comes at a trade-off, affecting recall, precision, and processing time, as mentioned earlier. Data reduction is achieved through powerful techniques of clustering and Similarity Reduction, which require considerable time to execute. However, this crucial step takes

place within cloud storage, resulting in considerable time savings when updating the model in the future and optimizing the cloud storage capacity. Furthermore, the frequency of the CBSE step is determined at long intervals. Surprisingly, the longer the interval, the lower is the time cost incurred during the processing step, emphasizing the efficiency gained through this approach. This intricate balance between data reduction, time optimization, and processing costs adds depth to the methodology, delivering a well-rounded and effective solution.

Minimizing the data size in model generation on the cloud offers several advantages:

- Reduced Storage Requirements.
- Faster Processing.
- Improved Scalability if needed.
- Enhanced Privacy and Security, because less data is transmitted and stored, organizations have more control over the data and can implement stricter security measures.

Overall, minimizing the data size in model generation on the cloud offers benefits, such as reduced storage requirements, faster processing, cost savings, improved scalability, and enhanced privacy and security. These advantages help optimize cloud resources, streamline operations, and enable organizations to derive insights and make informed decisions more efficiently.

A comparison of the results is presented in Table 5. [3] Measures are shaded. We can see that the accuracy percentages always outperform the related work measure, and the time required by most of the algorithms is only 2.495% of the related work time. Fig 9 shows that the CBSE had the best accuracy. The conclusion of the comparison in Table 6 is that the same evaluation measure is used = [(CBSE measure – [3] measure) / [3] measure], all time measures increased but the ANN and RF not by a little difference with a maximum of 5 s, because different platforms we run with, for precision there is an obvious improvement but ANN and GNB, not that is because these algorithms already need a huge data size because of the tradeoff with accuracy. However, DT improved with all metrics. For LR and KNN, the recall decreased by a small percentage, because some positive class records were eliminated.

Table 5. 6 Algorithms with Different Measure Percentages with [3]

Algorithm	**Accuracy**		**Precision**		**Recall**		**F1-Score**		**Time (In Seconds)**	
LR	92%	85%	84%	72%	60%	79%	65%	74%	0.1	8.2
DT	91%	79%	72%	70%	74%	71%	73%	70%	0	2.2
ANN	93%	85%	79%	82%	72%	74%	75%	77%	40	35
RF	95%	85%	92%	80%	77%	71%	82%	74%	1.8	1.4
KNN	92%	81%	82%	76%	63%	72%	68%	74%	0.3	11
GNB	85%	73%	64%	65%	73%	68%	67%	65%	0	0.2

Table 6. 6 Algorithms with Different Measure Metrics VS the Related Work 1 Using the Performance Equation

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Time**
LR	+9%	+17%	-23%	-12%	-99%
DT	+15%	+4%	+4%	+4%	-98%
ANN	+9%	-4%	-2%	-2%	+15%
RF	+12%	+15%	+8%	+11%	+27%
KNN	+14%	+9%	-12%	-8%	-98%

GNB	+17%	-1%	+7%	+3%	-93%

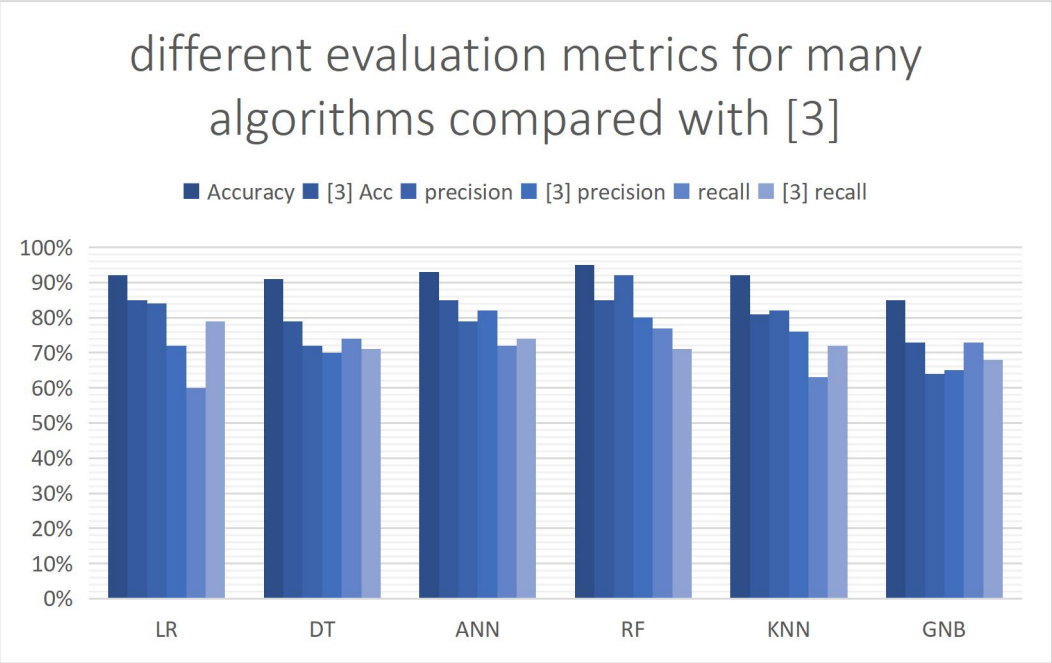

Figure 9. The Six Algorithms with 6 Evaluation Metrics Compared to [3].

5. Conclusion

One of the main challenges in IoT data analytics is overfitting, which occurs when a model is overly complex and memorizes the training data instead of learning from it. This can lead to deficient performance when the model is applied to new data. To address this problem, the proposed cluster-based-similarity elimination algorithm (CBSE) includes a preprocessing phase to remove duplicates and optimize data size before the classification model is applied. By reducing the amount of data and improving the quality of the remaining data, CBSE can improve the accuracy of the classification model while mitigating the risk of overfitting.

Experiments with different ML algorithms show that CBSE outperforms traditional approaches and related study techniques. The proposed methodology achieved a 59% reduction in execution time compared with the other approaches and a 91% reduction compared with the traditional approach. The focus on the classification task, which is crucial in many IoT applications, led us to develop a preclassification phase that can be used for real-time IoT data analytics. This technique has the potential to be explored further for prediction over a longer period and in different domains.

In summary, this research presents a novel approach to optimize data size and achieve accurate results with fast execution time, while also addressing the issue of overfitting. By leveraging CBSE, we can effectively handle real-time IoT data analytics using advanced methodologies that differ from conventional analysis techniques.

References

[1] R. Laing, R. Halsey, D. Donohue, C. Newman and A. Cashin, "Application of a model for the development of a mental health service delivery collaboration between police and the health service," *Issues in mental health nursing*, vol. 30,no. 5, pp. 337-341, 2009.

[2] A. Baldominos, A. Cervantes, Y. Saez and P. Isasi, "A comparison of machine learning and deep learning techniques for activity recognition using mobile devices," *Sensors*, vol. 19, no. 3, pp. 521, 2019.

[3] M. Vakili, M. Ghamsari, and M. Rezaei, "Performance analysis and comparison of machine and deep learning algorithms for IoT data classification," *arXiv preprint arXiv:2001.09636*, 2020.

[4] J. Suto, S. Oniga, C. Lung and I. Orha, "Comparison of offline and real-time human activity recognition results using machine learning techniques," *Neural computing and applications*, vol. 32, pp. 15673-15686, 2020.
[5] S. K. Bhoi, "Prediction of diabetes in females of pima Indian heritage: a complete supervised learning approach," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol.12, no.10, pp.3074-3084, 2021.
[6] V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, "Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms," *Neural Computing and Applications*, vol.35, no.22, pp.16157-16173,2023.
[7] G. Ngo, R. Beard and R. Chandra, "Evolutionary bagging for ensemble learning," *Neurocomputing*, vol. 510, pp.1-14, 2022.
[8] H. S. Obaid, S.A. Dheyab and S.S. Sabry, "The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning," *In 2019 9th AnnualInformation Technology, Electromechanical Engineering and MicroelectronicsConference (IEMECON)* , pp. 279-283,2019.
[9] I. Rish, "An empirical study of the naive Bayes classifier," *In IJCAI 2001 workshop onempirical methods in artificial intelligence*, vol. 3, no. 22, pp. 41-46, 2001.
[10] H. Naz, and S. Ahuja, 2020, "Deep learning approach for diabetes prediction using PIMA Indian dataset," *Journal of Diabetes & Metabolic Disorders*, vol.19, no.1, pp.391-403, 2020.
[11] C. M. Bishop and N.M. Nasrabadi, "Pattern recognition and machine learning,"*New York: springer*, vol. 4, no. 4, pp. 738, 2006.
[12] W.Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14-23, 2011.
[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5-32, 2001.
[14] N. S. Altman "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175-185, 1992.
[15] Trisha waghmare. "weatherAUS",Kaggle.com. [Online]. Available: https://www.kaggle.com/datasets/trisha2094/weatheraus. [Accessed: 04- Oct- 2022].
[16] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Computers & Security,* vol. 94, pp. 101863, 2020.
[17] Shafiq, M., Tian, Z., Bashir, A. K., Du, X., and M. Guizani, "CorrAUC: a Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine Learning Techniques," *IEEE Internetof Things Journal,* pp. 1-1, 2020.
[18] Xie W, Chi Y, Wang L, Yu K, & Li W., "A Novel Feature Selection Method Based on Binary Differential Evolution and Feature Subset Correlation for Microarray Data,"*InformationS ciences,* vol. 565, pp. 278-305, 2021.
[19] S. Su, Y. Sun, X. Gao, J. Qiu, & Z. Tian, "A Correlation-Change Based Feature Selection Method for IoT Equipment Anomaly Detection," *Applied Sciences*, vol. 9, no. 3, pp. 437, 2019.
[20] Kale, A. P., & Sonavane, S. P. "IoT based Smart Farming : Feature subset selection for optimized high-dimensional data using improved GA based approach for ELM,"*Computersa nd Electronics in Agriculture*. vol. 161, pp. 225-232, 2019.
[21] Gahar R. M., Arfaoui O., Hidri M.S. And Hadj-Alouane N.B., "A Distributed Approach for High-Dimensionality Heterogeneous Data Reduction," *IEEE Access*, vol. 7, pp. 151006-151022, 2019.
[22] I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, *Artificial Neural Networks*. Cham: Springer International Publishing, 2017.