Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences

Araştırma makalesi / Research article

www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

IoT-based fire detection: A comparative study of machine learning techniques

IoT-tabanlı yangın tespiti: Makine öğrenmesi tekniklerinin karşılaştırmalı çalışması

Ahmet Aytuğ Ayrancı[1,*], Burcu Erkmen[2]

[1] *İstanbul Kültür University Electrical and Electronics Engineering Department, 34158, İstanbul, Türkiye*
[2] *Yıldız Technical University Electronics and Communications Engineering Department, 34220, İstanbul, Türkiye*

Abstract

Fires that cannot be detected quickly become uncontrollable. The fires that start to spread uncontrollably pose a significant danger to humans and natural life. Especially in public and crowded areas, fires can lead to possible loss of life and massive property damage. Because of this, it is necessary to detect fires as accurately and quickly as possible. Smoke detectors used with Internet of Things (IoT) technology can exchange data with each other. In this study, data collected from two different types of IoT-based smoke detectors were processed using machine learning algorithms. The k-Nearest Neighbor (k-NN), Multilayer Perceptron (MLP), Radial Basis Function (RBF) Network, Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and Logistic Model Tree (LMT) algorithms were used. The data obtained from the smoke detectors were processed using machine learning algorithms to create a highly successful model design. The aim of the study is to design an artificial intelligence-based system that enables the early detection of fires occurring both indoors and outdoors.

Keywords: Machine learning, Fire detection system, IoT-based systems, K-fold cross validation

Öz

Hızlı bir şekilde tespit edilemeyen yangınlar kontrolsüz hale gelmektedir. Kontrolsüz biçimde yayılmaya başlayan yangınlar ise insan hayatına ve doğal yaşama büyük tehlike oluşturmaktadır. Özellikle halka açık ve kalabalık olan alanlarda başlayan yangınların olası can kayıplarına ve büyük maddi hasarlara yol açtığı görülmektedir. Bu nedenle yangınları mümkün olduğunca doğru ve hızlı bir şekilde tespit etmek büyük önem taşımaktadır. Nesnelerin İnterneti (IoT) teknolojisi ile birlikte kullanılan duman detektörleri birbirlerine veri akışı gerçekleştirebilmektedir. Bu çalışmada IoT-Tabanlı iki farklı tür duman detektöründen toplanan veriler makine öğrenmesi algoritmaları kullanarak işlenmiştir. Çok Katmanlı Algılayıcı (MLP), K-En Yakın Komşu (K-NN), Radyal Tabanlı Fonksiyon (RBF) Ağları, Naive Bayes (NB), Karar Ağacı (DT), Rastgele Orman (RF) ve Lojistik Model Ağacı (LMT) algoritmaları kullanılmıştır. Duman detektörlerinden elde edilen veriler makine öğrenmesi algoritmalarında işlenerek yüksek başarıya sahip bir model tasarımı sağlanmıştır. Çalışma sonucunda hem kapalı alanlarda hem de dış mekanlarda oluşan yangınların erken tespitinin mümkün olacağı bir sistem tasarımı hedeflenmektedir.

Anahtar Kelimeler: Makine öğrenmesi, Yangın tespit sistemi, IoT-tabanlı sistemler, K-katlı çapraz doğrulama

1 Introduction

Establishing a highly accurate fire safety system is necessary to prevent potential disasters in living spaces. Otherwise, major disasters can occur when an uncontrollable fire breaks out. The faster the fire is detected, the higher the probability of bringing it under control. Uncontrolled fires can result in the loss of life and property. With the latest advancements in IoT technology, it has become easier to establish a comprehensive fire detection system [1]. The most commonly used devices in fire detection systems are smoke detectors. These detectors, integrated with IoT technology, can share the data they receive with each other and the control center. By gathering information from multiple sensors, it is possible to achieve more accurate fire detection.

There have been numerous studies on disaster management and prevention in recent years [2-5]. Disasters pose a threat to human life and can result in the loss of lives and property if appropriate precautions are not taken. Among these disasters, fire is regarded as one of the most significant threats to human life and inhabited spaces [6]. If a fire is not detected early, it can cause extensive damage. Therefore, there have been many studies in the literature focusing on the early detection of fires [7-9]. These studies often revolve around smoke detection as a means to identify the onset of a fire. Smoke presence is typically determined either through image processing or by utilizing data obtained from smoke detectors [10-11]. Sensor-based methods detect changes in air temperature and smoke concentration to initiate early fire warnings. Automatic fire detection systems should be

* Sorumlu yazar / Corresponding author, e-posta / e-mail: ahmetayranci58@gmail.com (A. A. Ayrancı)
Geliş / Recieved: 06.03.2024 Kabul / Accepted: 20.08.2024 Yayımlanma / Published: xx.xx.20xx
doi: 10.28948/ngumuh.1444349

implemented to achieve faster detection and response to fires in indoor and outdoor areas. It is crucial to minimize false alarms in the fire detection system by considering various parameters. Information regarding fire incidents is collected using sensors or cameras.

There are two types of smoke detectors commonly used. These detectors are photoelectric smoke detectors and ionization smoke detectors. The features of these smoke detectors are explained in the following parts of the study. While detecting fire with smoke detectors, the source of the smoke must be determined correctly. False fire alarms can occur because smoke detectors have problems distinguishing whether the smoke source is a fire or another source [12]. To prevent false fire alarms, many parameters are used in the dataset to determine that the source of smoke is fire. False fire alarms cause time and financial losses. To avoid such losses, the fire detection system established must be reliable.

Seven classification algorithms were used for the analysis of the collected data: Multilayer Perceptron (MLP), Radial Basis Function (RBF) Network, k-Nearest Neighbor (k-NN), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and Logistic Model Tree (LMT). In the literature, Artificial Intelligence-based studies are becoming widespread in the early detection of disasters. Various Artificial Intelligence-based studies have also been conducted on fire detection systems. Image processing has been widely utilized in fire detection studies. In [13], the authors examined the color palette of the images captured from the suspected fire area for fire detection using the SVM algorithm. They employed the inter-frame technique to differentiate the suspected fire area from the background and extracted flame color moment features and texture features as inputs to the defined network. In the study presented in [14], the authors proposed the use of security camera systems in residential buildings for fire detection. They utilized Convolutional Neural Networks and Deep Learning to classify the images obtained from these camera systems. In [15], the authors investigated the combined use of image processing and sensor-based fire detection systems. They analyzed images captured from IP cameras along with data collected from smoke detectors, aiming to create a more reliable fire detection system. In [16], a fire detection system was designed using the Trend Predictive Neural Network model and data from multiple smoke detectors. The authors demonstrated that this model operated at a higher speed compared to conventional Neural Networks.

In this study, the Smoke Detection dataset, which was created by collecting data from IoT sensors in different fire scenarios, was used. The smoke Detection dataset includes 15 unique features derived from IoT smoke detectors. The 15 features represent collected sensor data, and 1 represents the classification result. These 15 features were reduced to 12 using data preprocessing steps. In this way, the aim is to make the model run faster and reduce the delay. The data were collected in fire scenarios occurring in open and closed areas. A comprehensive data set was created by collecting samples with a frequency of 1Hz under different fire scenarios. An artificial intelligence-based fire detection system can be developed using data collected in both indoor and outdoor fire scenarios. The accuracy of the fire detection system has been investigated using classification ML algorithms on the dataset. Results show that the RF algorithm provides the highest fire detection performance with a high accuracy rate of 99.98%. The NB algorithm achieved the lowest accuracy of 79.2%. The RF algorithm combines multiple decision trees with ensemble learning.

2 Materials and methods

This section discusses IoT smoke detectors, features in the dataset, and data preprocessing steps. Also, information about the seven classifier ML algorithms used is explained.

2.1 Smoke detectors

Smoke detectors are electronic devices used to detect smoke for possible fires. It is used for early detection of fires and to prevent possible loss of life in work areas and settlements. Smoke detectors sense smoke, which is the most common indicator of fires. However, since not all smokes are a sign of fire, it is important to note more than one parameter in fire detection. There are two types of commonly used smoke detectors. These detector types are Photoelectric and Ionization smoke detectors. In the smoke detection dataset, a more successful fire detection system design aimed at combining data from many sensors with sensor fusion techniques.

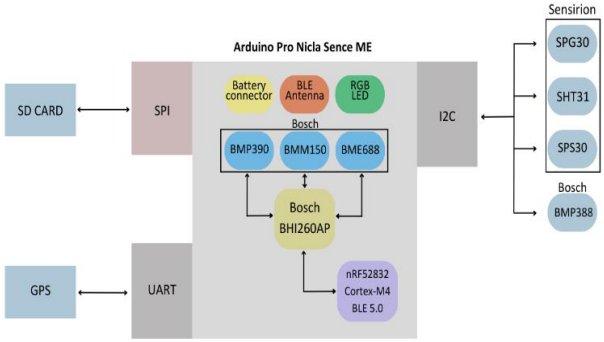

Figure 1. Block diagram of smoke sensor

In the smoke detection dataset, a more successful fire detection system design aimed at combining data from many sensors with sensor fusion techniques. Additionally, Bosch BMP388, Sensirion SPS30, Sensirion SHT31, Sensirion SPG30, and GPS sensors were used externally. The primary sensing component in the smoke detector is the Sensirion SPS30. The SPS sensor works similarly to photoelectric smoke detectors by measuring particles in the air. Particular matter (PM) in the dataset; PM1.0, PM2.5, and number concentration (NC): NC0.5, NC1.0, and NC2.0 outputs are given by the SPS sensor. Based on sensor fusion, the system is more cost-effective than the fire detection systems focused on image processing currently in use. Expected to have high performance, especially in indoor environments, it is anticipated to significantly enhance the performance of fire detection systems when used alongside existing systems at a low cost. The sensors utilized are highly affordable and accessible. The overall cost of the system is economical compared to a camera-based system.

A photoelectric smoke detector uses an infrared, ultraviolet, or visible light source to detect the presence of smoke. A photoelectric detector senses scattering light when the smoke enters the detector chamber and reaches the photosensor. The light emitted from the detector's photosensor is affected by the particles and dust in the smoke. The smoke detector generates an alarm when the light intensity coming from the photosensor falls below the specified threshold level. Photoelectric smoke alarms are usually more sensitive to smoldering fires.

An ionization smoke detector contains a small amount of radioactive material between two electrically charged planes. These detectors ionize the air using the radioactive material they have. The ionized air causes the electric current to flow between the plates. If the smoke comes into the detector chamber, the electric current flow is disturbed, and the detector starts to sound the alarm. Ionizing smoke detectors are often more sensitive in detecting flaming fires.

2.2 Dataset information

In this study, the Smoke Detection dataset created with the help of IoT smoke detectors was used. It is aimed to design an AI-Based IoT fire detection system using the smoke detector dataset shared over the Kaggle data repository [17]. The dataset contains 16 features, and these features are all numeric. These data in the dataset were collected under different fire scenarios from IoT smoke detectors. Three of these features do not affect the model. These features are timestamp, index and index count. The features that had no impact on the models were removed from the dataset through feature reduction method.

Since the index and index count variables specify the same property, both variables are excluded from the dataset. Also, the timestamp variable has a low weight on the models. There is no missing data in the dataset. The dataset contains information about air temperature, humidity, amount of CO2, organic compounds, and other gases. Five records are given in Table 1 to clarify the dataset. The 13 features used in the dataset show the air temperature, air humidity (%), volatile organic components (ppb), eCO2 concentration (ppm), Raw H2, Raw Ethanol, Air Pressure (hPa), Particle size, and particle concentration. In the fire, class used as output data, "0" indicates no fire, and "1" indicates fire. Fire indicates the output variable in the data set. The distribution of fire variable samples in the dataset is shown in Figure 1.

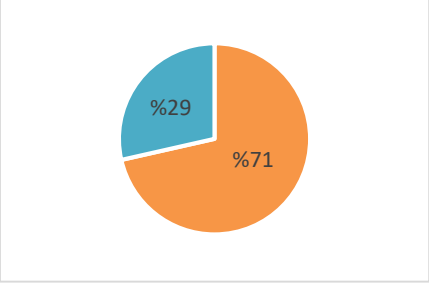

Figure 2. Distribution of fire feature in the dataset

There are 44757 fire and 17873 no fire examples in the fire variable used as the classifier variable. As given in Figure 1. the distribution in the dataset indicates %71 fire class and %29 no fire class. Ideally, the two classes should have equal amounts of data. Class imbalance is a common problem in many real-world datasets where the number of samples in one or more classes is much smaller than the others. To address this issue, two widely used techniques are undersampling and oversampling. Undersampling involves reducing the size of the majority class(es) to match the size of the minority class(es), while oversampling involves increasing the size of the minority class(es) to match the size of the majority class(es).

Undersampling can be advantageous for machine learning models that are sensitive to class imbalance, as it helps create a more balanced dataset, thereby potentially improving their performance. However, undersampling can also lead to the loss of valuable information and may not work well if the majority class(es) are already sparse. On the other hand, oversampling can improve the representation of the minority class(es) in the dataset, which can lead to better classification accuracy and generalization performance of machine learning models. Various oversampling methods have been proposed, including the synthetic minority oversampling technique (SMOTE), which creates new synthetic samples for the minority class(es) based on nearest-neighbor interpolation. However, it is important to note that oversampling can potentially introduce noise or bias into the dataset if the artificially generated samples do not accurately represent the underlying distribution of the minority class(es). The study did not employ undersampling or oversampling techniques for the fire detection task, as the researchers considered the amount and quality of the available data to be important factors. Additionally, the researchers decided against generating synthetic data due to concerns about potential impacts on the results.

2.3 Data preprocessing

Data preprocessing is the process of preparing raw data and fitting it into an ML model. Data preprocessing converts the dataset into a suitable format to work on. It is the first process to do when creating an ML model. Performing this process correctly ensures the model works with high performance. It is desired that the dataset used while ML modeling is clean and there are no outliers [18]. Otherwise, obtaining optimum performance in the modeling will not be possible.

Removing null and unnecessary data from the dataset can enhance the speed and effectiveness of ML models. While determining which data to clean, the impact of each variable on the output is carefully assessed. Fortunately, the dataset utilized in this study does not contain any null data. While the original dataset consisted of 16 variables, two of them, namely CNT and index, were excluded through data preprocessing as they did not contribute significantly to the dataset. This decision was based on the realization that CNT and index represented the same counting operation, resulting in redundant data. As a result, these variables were eliminated, resulting in a streamlined dataset.

Table 1. 5 example data from the dataset

Temperature	Humidity	TVOC	eCO2	H2	Ethanol	Pressure	PM1.0	PM2.5	NC0.5	NC1.0	NC2.5	Fire
22.105	54.36	32	437	12552	19637	939.809	0.0	0.01	0.01	0.012	0.006	0
22.725	54.45	18	400	12593	19669	939.806	0.18	0.4	0.67	0.448	0.214	0
17.38	53.66	6	400	13195	20120	939.62	0.79	0.83	5.47	0.853	0.019	0
9.829	49.86	1320	404	12992	19404	938.814	1.75	1.82	12.04	1.877	0.042	1
26.29	50.77	1258	400	12999	19415	928.862	1.87	1.94	12.88	2.009	0.045	1

2.4 Classification algorithms

Classification algorithms assign the output feature to one of the specified classes based on the inputs in the dataset. In this study, fire detection analysis was performed by processing 13 feature in the dataset. Seven classification algorithms were used for the Artificial intelligence-based fire detection system. These algorithms are Multi-Layer Perceptron (MLP), k-Nearest Neighbor (k-NN), Radial Basis Function (RBF), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and Logistic Model Tree (LMT).

K-fold cross-validation is applied to the study to ensure unbiased observations. It is a data partitioning strategy to ensure model performance. The cross-validation technique divides the dataset into k parts. In the k-fold cross-validation technique, k-1 folds are used for training, and one-fold is used for performance evaluation in the model. To make unbiased observations dataset was divided randomly. Choosing a k number high in the k-fold cross-validation technique makes it easier to design an unbiased model [19]. However, choosing the k number high increases the computational complexity and causes slower modeling.

Hyperparameter optimizations were conducted to fine-tune various parameters in the classification algorithms. For instance, in the Random Forest (RF) algorithm, the "total number of trees" was tuned to achieve optimal performance. Similarly, the k parameter that shows the number of neighbors in the K-NN algorithm was adjusted to find the most suitable value. In the case of the Multilayer Perceptron (MLP) algorithm, the "hidden layer number" was optimized to enhance the model's effectiveness. These hyperparameter optimizations aimed to maximize the accuracy and overall performance of the classification algorithms.

2.4.1 Multi-layer perceptron

MLP is one of the basic algorithms widely used in classification problems. As the name indicates, it is an artificial neural network algorithm consisting of many layers. The MLP algorithm consists of an input layer, an output layer, and one or more hidden layers. The MLP algorithm can also work on complex datasets using non-linear functions [20].

The input data is fed into the input layer of the MLP model. Within this layer, the data is processed and transmitted to the hidden layer using activation functions for preprocessing. The sigmoid function generally uses as an activation function. The number of hidden layers in the MLP structure is determined as a parameter and can be selected as single or multiple layers. The data flows from the input layer to the output layer, resembling a feed-forward network. During this process, weights are assigned to each data point using the designated activation functions. The backpropagation learning algorithm is used to train MLP neurons. This learning algorithm works by testing backward errors starting from the output nodes [21].

MLP is one of the classifier algorithms where hyperparameter tuning has a significant impact. The number of hidden layers, momentum, and learning rate parameters are the main optimized parameters in the MLP algorithm. Hyperparameter optimization has a significant impact on the MLP algorithm. Although the MLP algorithm seems to have a simple structure, it needs a lot of time in the modeling and training phases. In MLP, the testing phase takes less time. The hidden layer and the number of neurons determine the complexity and speed of the algorithm. The accuracy of the model can be increased by optimizing the parameters in the algorithm structure.

2.4.2 K-nearest neighbor

K-NN, initially introduced by Fix and Hodges in 1951 for pattern problems, was further developed by Cover and Hart in 1967 [22]. It is a straightforward machine learning algorithm known for its simplicity and interpretability. Due to its uncomplicated structure, K-NN finds application in both classification and regression problems. K-NN is classified as a supervised learning algorithm, but it distinguishes itself from other ML algorithms by not having a dedicated training stage. As a result, K-NN is considered a lazy learning algorithm, relying on memorization rather than conventional learning methods.

The K-NN structure predicts the class of a given data sample based on the concept of feature similarity. This similarity is determined by selecting the value of k, which is a crucial factor influencing the accuracy of the K-NN model. In the K-NN model k parameter specifies the number of nearest neighbors to check. It is essential to carefully choose the value of k to prevent overfitting or misclassification. Selecting a small k value may lead to overfitting while choosing a large k value increases the risk of misclassification. Therefore, determining the optimal k value is crucial in the K-NN structure. Depending on the problem at hand, a higher k value is preferred to reduce overall noise. However, as the k increases, the probability of misclassification also rises. To determine the optimal k number cross-validation method can be utilized. In this study, a k value of 3 was selected as the optimal parameter.

Euclidean and Manhattan distance functions are widely used while calculating the distance in the K-NN structure.

The average value is assigned to the unknown nodes when using the Euclidean distance function. The Euclidean distance function tries to predict missing nodes from neighboring node values. The formula for the Euclidean distance function is given below.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

In Equation (1), n represents number of data points, d represents distance, x_i indicates the incoming data, and y_i indicates the selected data. In the Manhattan distance formula, the distance between two nodes is the sum of the absolute value of the difference in the cartesian coordinates. The Manhattan distance formula is defined in Equation (2).

$$d(x,y) = \sum_{i=1}^{n}(x_i - y_i) \quad (2)$$

Manhattan distance formula uses same abbreviations with Euclidian distance formula. In Equation (2), n represents number of data points, d represents distance, xi indicates the incoming data, and yi indicates the selected data. Euclidian and Manhattan distance functions can be used for continuous data. If the dataset consists of categorical data, Hamming distance function should be used. In case of numerical and categorical data exist together, then numerical data should be standardized.

2.4.3 *Radial basis function*

RBF network is an Artificial Neural Network (ANN) developed by Moody and Darken in 1989 [23]. The design of the RBF network was inspired by biological nerve cells. RBF networks, which are artificial neural networks model, have a different structure from other artificial neural network models. In comparison to other artificial neural networks, RBF networks have a different structure. RBF networks have an input layer, a single hidden layer, and an output layer. The point that distinguishes RBF networks from other artificial neural networks is the operation happens in this single hidden layer. The data converted to feature vectors in the hidden layer are extracted to higher dimensions with the help of non-linear transfer functions. Then the classification process is performed in the output layer.

2.4.4 *Naïve bayes*

Naïve Bayes (NB) classifier is a mathematical ML algorithm based on conditional probabilities in estimation. As mentioned in the name of the algorithm, it is based on the Bayesian formula used in conditional probability calculations [24]. NB classifier can do both numerical and categorical classification. NB provides high accuracy in complex classification problems. The main advantages of the algorithm are that it can quickly model complex problems and require very little data for training. NB calculates the probability of each state for an element and classifies it according to the highest probability value. Algorithm considers each feature in the dataset as independent from the other. Since the features are assumed to be independent of each other, the relations between the variables cannot be modeled.

One of the crucial problems with the NB algorithm is zero frequency. It means that the desired sample from the model cannot be found in the dataset. Zero frequency occurs when any condition having zero probability in the whole multiplication of the likelihood makes the complete probability zero. The simplest way to solve this is to eliminate this possibility by adding the minimum value to all values. This solution is called the Laplace Estimator.

2.4.5 *Decision tree*

DT is an ML algorithm widely used in both classification and regression problems in supervised learning. DT is a simple algorithm that applies some rules when making decisions. Tree-based ML algorithms are the most widely used supervised learning algorithms. The DT algorithm can be used for classifying both numerical and categorical data. The structure of DT is similar to the thought system of people. DT algorithm defines the dataset by dividing it into small decision nodes. A set of rules is applied at the decision nodes to create a structure that divides into different branches [25].

The first node in the algorithm is called the root node. From this node, the downstream decision nodes are designed then the leaf nodes follow. The DT algorithm is structurally in the form of a tree, and probabilities of all classification sets are defined in its branches. Leaf nodes are the final decision points in the DT algorithm. Splitting the data in the DT algorithm has a significant impact on the accuracy of the model. As a result of dividing the data into sub-nodes, the model becomes more homogeneous. DT uses various techniques to split a decision node into two or more nodes. The most used techniques in the DT algorithm are entropy and information gain. Entropy can be defined as a measure of randomness and uncertainty. If the samples are ordered and homogeneously separated, the entropy is zero. Therefore, it is desirable to arrange the data in such a way that the entropy is minimal. Information gain is subtracting all entropy after dividing a dataset over a feature. The importance of the feature increases if the entropy value is small.

2.4.6 *Random forest*

In 2001 Leo Breiman developed Random Forest (RF) algorithm [26]. It utilizes an ensemble learning technique that combines many classifiers to solve complex problems. RF algorithm contains many decision trees in its structure. As the number of decision trees used in the algorithm increases, the possibility of creating a more successful model increase. One of the important differences between it and the DT algorithm is that finding the root node and splitting the nodes in the RF algorithm is random. This way, there is no need to use entropy and information gain techniques.

Overfitting, which is the most significant problem of DT, decreases in the RF classifier. RF eradicates the limitations of the DT algorithm. It reduces the overfitting of datasets and increases accuracy. Overfitting can be overcome by using enough decision trees in the RF algorithm. Another advantage of the RF algorithm is that it displays the weights of the classes. RF needs less data and preparation time for training. It is also successful on datasets with noisy and

outlier samples. The RF algorithm utilizes the GINI index to assess the effectiveness of trees and branches, which measures the success rates of the samples assigned to each node. A low GINI index value indicates the homogeneous distribution of the dataset and the successful creation of branch nodes. The GINI index in the RF algorithm is used similarly to the information gain technique in the DT algorithm [27].

RF is a very suitable algorithm for hyperparameter optimization. It has several hyperparameters to be set. These parameters are the number of trees, the number of random features, the number of nodes, and the splitting rule. The high number of trees enables to build of a successful model. The selection of the number of random features is one of the most crucial parameters. This hyperparameter is hard to pick without experimenting. Choosing a low number of features means that fewer features are considered when splitting data. Adjusting the number of nodes influences reducing overfitting in the model. RF uses GINI and Entropy techniques for data splitting. Hyperparameter tuning is essential for RF algorithms.

2.4.7 Logistic model tree

Logistic Model Tree (LMT) is a supervised learning algorithm classification model. It has high accuracy in binary and multiple classification problems. LMT combines the features of Logistic Regression (LR) and DT algorithms. Unlike the decision trees method, the leaf nodes in the LMT algorithm have logistic regression functions of related attributes as well as class labels [28]. Logit Boost enables leaf nodes to utilize logistic regression functions. However, logistic regression has a complex structure. Due to its complex structure, the construction of the tree model takes time.

2.4.8 K-fold cross validation

In classification problems, the initial step is to divide the dataset into training and testing sets. It is crucial to be aware that challenges may arise during this process of splitting the dataset into training and testing sets. One potential issue is the uneven distribution of classes, which can lead to a biased model. The K-fold cross-validation technique's goal is to create unbiased observation sets. K-fold cross-validation divides the dataset into k sets [29]. In K-fold cross-validation, all subsets are treated as equal in terms of size and quality. One of the divided subsets is designated as the test set, while the remaining k-1 subsets are used for training the data. This validation process is repeated k times, with each subset taking turns as the test set.

Increasing the number of k in the K-fold cross-validation technique increases the possibility of creating an unbiased model. However, increasing the k number also causes a loss of computation and time. Using more than one k value provides us to see the performance of the k-fold cross-validation technique. Thus, it is possible to find the optimum value of k. As a result of K-fold cross-validation, the success of the designed model can be seen by examining the performance metrics. K-fold cross-validation shows unbiased prediction accuracy. Cross-validation can be utilized to identify the presence of overfitting in a model.

3 Performance evaluation

A classification study was conducted on the Smoke Detection dataset using the Python 3.11 program. The study utilized seven machine learning classifier algorithms. The main objective of this research was to create an efficient fire detection system by implementing ML algorithms on the data collected from IoT smoke detectors. The performance of the designed system was evaluated using various ML performance metrics.

3.1 Performance metrics

In this section, we will provide a brief explanation of the performance metrics used to evaluate the performance of the models employed. Performance metrics have a significant role in assessing the effectiveness of ML models. These metrics are utilized to measure the success of classifier models on the dataset. The primary performance metrics employed in classifiers include accuracy, precision, recall, F-score, and kappa statistics. These metrics indicate the percentage of correct classification predictions made by the model. When evaluating the performance of the classifier, key parameters to consider are True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These parameters are essential for assessing both performance and error metrics accurately.

3.1.1 Accuracy

The most crucial performance metric in classification problems is accuracy, which indicates the rate of correct classifications. Accuracy measures not only TP classifications but also TN classifications [30]. It can be succinctly stated as the correct classification divided by the total classification. Equation (3) states the accuracy value of the model.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

Accuracy is the first performance metric to look at in classification problems. Higher accuracy value usually indicates the model's success. In some cases, achieving high accuracy values can be attributed to overfitting. In such situations, it is essential to examine the presence of overfitting.

3.1.2 Precision

Precision is a performance metric employed in both classification and regression problems. It reflects the accuracy of correctly made classifications. Precision is a metric that measures the ratio of TP samples to all positive samples. Precision measures the positive classification success of the model. It can be obtained as shown in Equation (4).

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

Precision has a significant impact when examining the performance of the model. TP samples are important for the fire detection system designed in this study. When the FP classification rate is high, the fire alarm may activate in the

absence of fire, causing unnecessary panic and disruption in both living and working areas. Also, it may cause a waste of time and intervention to other possible fires by the firefighters in charge of fire extinguishing. Because of this, the Precision metric wanted to be as high as possible in the model created.

3.1.3 Recall

Recall indicates how accurately the positive samples were measured. The recall metric has similarities with the precision metric. Precision and recall are performance metrics used for pattern recognition and classification in machine learning. These performance metrics are crucial for building a good ML model. When determining the recall of a model, it focuses solely on positive values while ignoring negative values. The recall metric reveals the effectiveness of correctly classifying instances in all fire-related scenarios. Recall can be calculated using Equation (5).

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

Recall helps us to see how accurately the relevant data in the dataset is classified. The recall metric demonstrates the accuracy of the fire alarm activation in the event of a fire. In this study, it can be stated that recall is the most critical performance metric, as it reflects the accuracy in fire-related cases.

3.1.4 F-score

The F-score metric is used to measure the accuracy value of the classifier model. The F-score metric uses precision and recall. Since these two metrics are not comprehensive on positive classifications, we can examine positive classified samples with the F-score metric. F-score takes the harmonic average of precision and recall metrics. The F-score can be found using Equation (6).

$$F - Score = \frac{Precision*Recall}{Precision+Recall} \quad (6)$$

F-score has the same weight as it is the average of the precision and recall metrics.

3.1.5 Kappa statistics

Kappa statistics works similarly to the accuracy metric in the classification model. It is used especially in multi-classification models because accuracy, precision, and recall metrics do not represent the model comprehensively. This metric works on a probability basis. The Kappa statistic value is calculated based on expected and observed probabilities. The value of the Kappa metric is one or less than one. A Kappa value close to 1 indicates that the model is successful. In general, kappa values exceeding 0.75 indicate a successful model, while values close to 0 suggest an unsuccessful model. If the Kappa value is 0 or less than 0, it is assumed that there is a random relationship between the raters, and the model is considered unsuccessful.

3.2 Error metrics

Error metrics are used for identifying failed classifications made by the model. To analyze the classification accuracy of the model, we should also investigate misclassifications. The error performance of the models was analyzed by examining the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics.

3.2.1 Mean absolute error

Mean Absolute Error (MAE) is a metric that measures the errors that occur in the model statistically. The MAE error level is directly related to the accuracy of the model. It shows the average difference between the true value of the classified variable and the predicted variable. MAE is more robust towards outliers than RMSE metrics since the changes in MAE are linear and, therefore intuitive [31]. While MSE and RMSE metrics perform better against large-scale errors, MAE analyzes errors linearly, so there is no performance change based on error level. The MAE provides a measure of the model's ability to predict actual values. A lower MAE indicates higher accuracy, while a higher MAE indicates lower accuracy. The MAE is a useful metric due to its simplicity and comparability across different models. However, it has a limitation in that it treats all errors equally, regardless of their magnitude. It means that both large and small errors carry the same weight in the calculation. For certain applications, this may not be desirable, and alternative metrics like the Root Mean Squared Error (RMSE) may be more suitable.

3.2.2 Root mean square error

The Root Mean Square Error (RMSE) measures the standard deviation of the error, providing insight into the dispersion of values. The standard deviation is employed in classifiers to detect outliers. When evaluating error performance, the RMSE value is utilized. To obtain the RMSE value, the variance value must first be determined, which represents deviations from the mean. The RMSE value is obtained by taking the square root of the calculated variance value. It is also equivalent to the square root of the Mean Square Error (MSE) metric. In ML models, the deviation of data from the mean line is utilized instead of variance. RMSE metric is employed to assess performance in both classification and regression models. A smaller RMSE value indicates a better-performing classification model, signifying a stronger fit of the model to the data.

3.3 Classification model's performance

A fire detection system has been designed with ML algorithms in IoT-based smoke detectors. The performance of accuracy, recall, and precision metrics in the ML models is significant for the system.

To the authors' knowledge, there is no study yet using this dataset. In this study, seven ML models were tested on the dataset using the Python Jupyter Notebook environment. While testing the models, the dataset was randomly divided into 80% training and 20% testing data. The dataset was divided in this ratio to obtain the most optimal result. Accuracy, Precision, and Recall results obtained in the models are given in Table 2.

Table 2. Accuracy, precision, and recall results of algorithms

Algorithms	Accuracy	Precision	Recall
MLP	96.02	93.00	96.00
K-NN	97.87	97.90	98.00
RBF	86.93	87.70	87.00
NB	81.50	82.40	81.50
DT	99.76	99.80	99.90
RF	99.98	99.90	100
LMT	98.95	99.00	99.00

The models' accuracy, precision, and recall values are shown in the Table 2. The results show that tree-structured algorithms perform better than other classifier models. The performance results of the models also show that the dataset is suitable for the used models. The RF model achieved the highest classification performance. Also, DT and LMT algorithms achieved similar performance to the RF model. This proves the suitability of tree classifier algorithms for the dataset. F-score and Kappa statistic metrics were also examined while comparing the performance of the models. These performance metrics verify the relative weight of precision and recall metrics and the quality of the model. F-score and Kappa statistic values of the algorithms are given in Table 3.

Table 3. F-score and kappa statistic results of algorithms

Algorithms	F-score	Kappa statistics
MLP	98.00	97.17
K-NN	99.15	97.90
RBF	89.20	72.90
NB	79.20	57.70
DT	99.88	99.82
RF	100	99.98
LMT	99.90	99.75

The results of the performance metrics for the designed models are presented in Table 2 and Table 3. The findings indicate that tree-based algorithms demonstrate high performance, while the NB algorithm exhibits the lowest model performance. However, despite its lower performance, the NB algorithm still achieves a high accuracy of 81.5%, suggesting its usability. Figure 3 provides a graphical representation of the accuracy, precision, and recall metric values for the models.

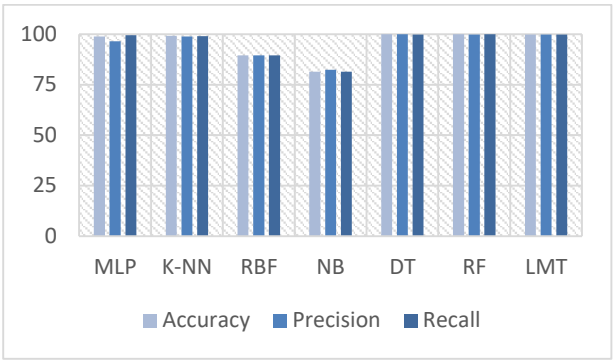

Figure 3. Performance metrics values of models

Error performance metrics in machine learning are crucial for assessing and improving the accuracy and robustness of models. These metrics provide quantitative measures of a model's prediction errors. These performance metrics are crucial for evaluating the effectiveness of designed ML models. When investigating the accuracy of the models, it is necessary to examine the performance metrics and error metrics together. Two error metrics were assessed to determine the rates of unsuccessful classification and which class had more classification errors. This study examines the MAE and RMSE error metrics to show the error performance of the models. Table 4 shows the MAE and RMSE values of the algorithms.

Table 4. Error performance of algorithms

Algorithms	MAE	RMSE
MLP	4.95	16.10
K-NN	2.40	12.50
RBF	18.10	29.75
NB	21.10	41.50
DT	0.08	0.18
RF	0.02	0.32
LMT	0.03	0.827

Table 4 indicates the error performance of the used algorithms. It is seen that tree-structured algorithms exhibit higher error performance.

Table 5. Accuracy values obtained using the K-fold cross-validation.

Algorithms	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
MLP	95.56	95.90	96.00	96.50	96.30	**96.58**	96.15	95.75	96.40
K-NN	97.65	97.85	98.00	98.00	98.10	98.15	98.15	**98.20**	98.10
RBF	87.50	86.50	86.10	87.40	**88.15**	86.70	87.65	87.00	87.00
NB	78.20	**78.40**	78.25	78.30	78.25	78.20	78.00	77.90	77.50
DT	99.84	99.86	99.89	99.90	99.97	99.95	**99.98**	99.97	99.95
RF	99.90	99.92	99.94	99.95	99.97	99.98	**99.99**	99.98	99.95
LMT	99.70	99.50	97.50	97.90	98.40	99.00	**99.50**	97.90	97.50

Since the tree-structured algorithms have better performance than other algorithms, the error performance of these algorithms is also expected to be higher than others. RF achieved the highest error performance among the used models, and NB showed the lowest error performance.

3.4 K-fold cross validation results

This section presents the test results using the K-Fold cross-validation technique to assess the performance of the models. The performance of the models was evaluated for various k parameters. Table 5 displays the accuracy values obtained from the k-fold cross-validation technique across different k parameters. Upon analyzing the results of the k-fold cross-validation, it can be inferred that the dataset exhibits independence and lack of bias. The results indicate that RF algorithms achieve the highest performance, whereas NB algorithms exhibit the lowest. These findings align with the results obtained from the model created by partitioning the dataset into train and test sets.

4 Results and discussion

This study aims to design a system that can detect fires early using AI-based IoT smoke detectors. It includes an examination of the dataset derived from IoT smoke detectors, followed by the application of data preprocessing steps. Data preprocessing steps were applied dataset to evaluate the quality of the data. Three features that do not affect the models are removed for faster processing speed and less complexity on the model. There is no missing data. The data values in the attributes shows a balanced distribution.

To the best of the authors' knowledge, the Smoke Detection dataset used in the study has not yet been used in any other studies in the literature. Therefore, we evaluated studies on similar datasets to analyze the study's performance improvement. In [15], the authors gathered data from various sensors and conducted a study on fire detection using Trend Predictive Neural Network (TPNN) and MLP algorithms. The TPNN algorithm achieved the highest accuracy of 99.7%. In our research, 99.98% accuracy is obtained with the RF algorithm. The training time of the RF model was seen as 1 second, while the testing time of the model was 0.02 seconds. The results obtained indicate that the models are appropriate for real-time usage.

This study aimed to achieve early fire detection by applying seven classification algorithms to a smoke detection dataset. The results demonstrate that the RF algorithm achieved the highest level of success, while the NB algorithm displayed the lowest performance. These findings hold significant implications for the design and implementation of IoT-based smoke detectors, indicating that ML models can improve accuracy and reliability. Upon thorough examination of the results, it becomes evident that AI-based fire detection systems have the potential to facilitate the early identification of fires, mitigating the potential for extensive damage in both indoor and outdoor environments. Implementation of such systems can effectively minimize the impact on human life and the environment. Future research endeavors will incorporate visual sensors in conjunction with smoke detectors to enhance reliability and precision. By integrating data from both smoke detectors and cameras and leveraging sensor fusion algorithms, it becomes feasible to develop a system that ensures more dependable and timely fire detection.

Conflict of Interest

The authors declare that there is no conflict of interest.

Similarity Rate (iThenticate): %12

References

[1] K. Mehta, S. Sharma, and D. Mishra, Internet-of-Things enabled forest fire detection system, 2021 Fifth International Conference on I-SMAC (IoT in Social Mobile Analytics and Cloud), pp. 20-23, Palladam, India, 2021.

[2] J. Lu, J. Guo, Z. Jian, X. Xu, Optimal Allocation of Fire Extinguishing Equipment for a Power Grid Under Widespread Fire Disasters, IEEE Access, vol.6, pp. 6382-6389, 2018. https://doi.org/10.1109/ACCESS.2017.2788893.

[3] Y. Hirohara, T. Ishida, N. Uchida and Y. Shibata, Proposal of a Disaster Information Cloud System for Disaster Prevention and Reduction, WAINA 2017, pp. 664-667, 2017.

[4] Ö. Doğan, O. Şahin, and E. Karaaslan, Digital twin based disaster management system proposal: DT-DMS. Journal of Emerging Computer Technologies, 1(2), 25-30, 2021.

[5] E. N. Soysal, H. Gürkan, and E. Yavşan, IoT Band: A wearable sensor system to track vital data and location of missing or earthquake victims. International Journal of Computational and Experimental Science and Engineering, 9(3), 213-218, 2023. https://doi.org/10.22 399/ijcesen.1317040.

[6] J. Qiu, J. Wang, T. He, B. Chen and X. Chen, Research on intelligent fire rating evaluation and rapid rescue plan optimization strategy, 2020 International Conference on Urban Engineering and Management Science (ICUEMS), pp. 446-453, 2020.

[7] K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, P. Yang, and S. W. Baik, Efficient deep CNN-based fire detection and localization in video surveillance applications. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(7), 1419-1434, 2018. https://doi.org/10.1109/TSMC.2018.2830099.

[8] A. Sungheetha and R. Sharma, Real time monitoring and fire detection using internet of things and cloud based drones. Journal of Soft Computing Paradigm (JSCP), 2(03), 168-174, 2020. https://doi.org/10.36548/jscp.2020.3.004.

[9] A. F. Önal, B. Ulver, A. Durusoy, and Erkmen, B., Intelligent wireless sensor networks for early fire warning system. Electrica, 2020. https://doi.10.26650/electrica.2019.19019.

[10] T. Çelik, H. Özkaramanlı, and H. Demirel, Fire and smoke detection without sensors: Image processing

based approach, 2007 15th European Signal Processing Conference, pp. 1794-1798, 2007.
[11] F. M. A. Hossain, Y. Zhang, C. Yuan, and C. Y. Su, Wildfire flame and smoke detection using static image features and artificial neural network, 2019 1st International Conference on Industrial Artificial Intelligence (IAI), pp. 1-6, 2019.
[12] K. Chen, Y. Cheng, H. Bai, C. Mou and Y. Zhang, Research on image fire detection based on support vector machine, 2019 9th International Conference on Fire Science and Fire Protection Engineering (ICFSFPE), pp. 1-7, 2019.
[13] N. A. Mwedzi, N. I. Nwulu and S. L. Gbadamosi, Machine learning applications for fire detection in a residential building, 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS), pp. 1-4, 2019.
[14] N. Chowdhury, D. R. Mushfiq and A. E. Chowdhury, Computer vision and smoke sensor based fire detection system, 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1-5, 2019.
[15] M. Nakıp and C. Güzeliş, Multi-sensor fire detector based on trend predictive neural network, 2019 11th International Conference on Electrical and Electronics Engineering (ELECO), pp. 600-604, 2019.
[16] M. Nakip, C. Güzelíş and O. Yildiz, Recurrent Trend predictive neural network for multi-sensor fire detection, in IEEE Access, vol. 9, pp. 84204-84216, 2021. https://doi.org/10.1109/ACCESS.2021.3087736.
[17] Kaggle. Smoke Detection Dataset. Available from: https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset September 04, 2022.
[18] A. A. Ayrancı, S. Atay and T. Yıldırım, Speaker accent recognition using machine learning algorithms, 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 1-6, Istanbul, Turkey, 2020. https://doi.org/10.1109/ASYU50717.20 20.9259902.
[19] J. D. Rodriguez, A. Perez and J. A. Lozano, Sensitivity analysis of k-fold cross validation in prediction error estimation in IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 3, 569-575, 2010. https://doi.10.1109/10.1109/TPAMI.2009.187.
[20] F. Murtagh, Multilayer perceptrons for classification and regression, Neurocomputing, 2(5-6), 183-197, 1991.
[21] M. K. Alsmadi, K. B. Omar, S. A. Noah and I. Almarashdah, performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks, 2009 IEEE International Advance Computing Conference, pp. 296-299, Patiala, India, 2009.
[22] T. Cover and P. Hart, Nearest neighbor pattern classification, in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967. https://doi.10.1109/TIT.1967.1053964.
[23] J. Moody and C. J. Darken, Fast learning in networks of locally-tuned processing units, Neural computation, vol. 1, no. 2, pp. 281-294, 1989. https://doi.10.1162/neco.1989.1.2.281.
[24] R. Irina. An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence, vol. 3, no. 22, pp. 41-46, 2001.
[25] S. R. Safavian, and D. Landgrebe, A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3), 660-674, 1991. https://doi.org/10.1109/21.97458.
[26] L. Breiman, Random forests. Machine learning, 45(1), 2001. https://doi.org/10.1109/COMST.2015.2494502.
[27] H. Han, X. Guo, and H. Yu, Variable selection using mean decrease accuracy and mean decrease gini based on random forest. In 2016 7th IEEE international conference on software engineering and service science, pp. 219-224, 2016.
[28] D. L. Gupta, A. K. Malviya, and S. Satyendra, Performance analysis of classification tree learning algorithms. International Journal of Computer Applications, 2012. https://doi.org/10.1007/978-3-319-03844-5_9.
[29] J. D. Rodriguez, A. Perez, and J. A. Lozano, Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE transactions on pattern analysis and machine intelligence, 32(3), 569-575, 2009. https://doi.org/ 10.1109/TPAMI.2009.187.
[30] S. Garcia, A. Fernández, J. Luengo, and F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing, 13, 959-977, 2009. https://doi.org/10.1007/s00500-008-0392-y.
[31] T. Chai, and R. R. Draxler, Root mean square error (RMSE) or mean absolute error (MAE) Arguments against avoiding RMSE in the literature. Geoscientific model development, 7(3), 1247-1250, 2014. https://doi.org /10.5194/gmd-7-1247-2014.