**Smart Fire Prevention by Forecasting Alarms Through Multi – Sensor Data Insights**

**ABSTRACT**

The project focuses on developing a smart fire prevention system by forecasting fire alarms using multi-sensor data insights. The primary goal is to leverage sensor data to predict fire alarm activation accurately, enabling timely interventions and reducing fire-related hazards. The project began with data collection from various sensors measuring environmental parameters relevant to fire detection. Initial exploratory data analysis (EDA) included visualizing the distribution of fire alarms and examining data quality, including checks for missing values and duplicate records. Duplicate entries were removed to ensure data integrity. The dataset consisted of both numerical and categorical features, which were preprocessed accordingly. Categorical variables were encoded using label encoding, and numerical features were standardized with feature scaling to improve model performance. To address the imbalanced nature of the dataset, resampling techniques were considered but not applied in the final modeling. The dataset was split into training and testing sets with an 80-20 ratio, enabling unbiased evaluation of the models. Two machine learning algorithms were implemented and evaluated: Logistic Regression and Random Forest Classifier. Logistic Regression was configured with L2 regularization and a very small inverse regularization parameter to avoid overfitting. The model achieved an accuracy of 71.3% on the test data, indicating a moderate ability to distinguish between alarm states. Random Forest Classifier, an ensemble method based on decision trees, was trained with 200 estimators and a maximum depth of 15. This model delivered a perfect accuracy score of 100%, demonstrating exceptional predictive capability on the given dataset. Both models’ performance was evaluated using precision, recall, F1-score, and confusion matrices, providing a comprehensive understanding of classification effectiveness. The Random Forest model’s superior accuracy suggests its strong suitability for real-time fire alarm prediction systems based on multi-sensor inputs. This work lays the foundation for deploying intelligent fire prevention systems that improve safety and emergency response.

**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

Fire incidents cause severe damage to life, property, and environment worldwide. In India, fire accidents have been increasing steadily, with thousands of casualties and significant economic losses each year. According to the National Crime Records Bureau (NCRB), thousands of fire-related deaths occur annually, mainly due to delayed detection and response. Traditional fire detection systems often rely on single-sensor alarms, which may produce false alarms or miss early warnings. The advancement in sensor technology and data analytics enables the integration of multiple sensor readings such as smoke, temperature, and gas concentration to enhance fire prediction accuracy. This project aims to utilize multi-sensor data and machine learning techniques to predict fire alarms proactively. Applications range from residential safety systems, industrial fire prevention, forest fire monitoring, to smart city infrastructure, making fire safety more reliable and automated.

**1.2 Problem Definition**

Before the application of machine learning, fire detection systems mostly depended on threshold-based alarms triggered by individual sensors, resulting in high false alarm rates and delayed detection. These systems lacked the ability to analyze complex sensor data collectively, causing inefficiencies in early fire prediction. Manual monitoring or simplistic rule-based systems could not adapt to varying environmental conditions or sensor noise. Consequently, fire emergencies were often recognized too late, leading to extensive damage. The absence of intelligent prediction models limited timely interventions, highlighting the need for automated, data-driven approaches to accurately forecast fire alarms and reduce false positives.

**1.3 Research Motivation**

The increasing frequency and severity of fire incidents motivate the development of smarter fire prevention mechanisms. Machine learning offers powerful tools to analyze vast multi-sensor data streams and identify patterns indicating imminent fire risks. This project explores leveraging these algorithms to enhance prediction accuracy and response time. Improving fire alarm systems not only saves lives and property but also optimizes emergency resource allocation. Motivated by advancements in IoT and AI, this research aims to create a predictive model that integrates multiple sensor inputs for robust fire alarm forecasting, supporting safer living and working environments.

**1.4 Need for the Project**

Current fire detection methods are often reactive rather than proactive, failing to provide timely alerts to prevent disaster escalation. False alarms cause unnecessary panic and resource wastage, while missed detections increase risk. With urbanization and industrial growth, fire hazards are escalating, demanding smarter detection systems. This project fills the gap by using multi-sensor data and machine learning to accurately predict fire alarms, reducing false positives and enabling early warnings. A reliable predictive system supports safer infrastructure, minimizes fire damage, and enhances public safety and emergency preparedness.

**1.5 Exploratory Data Analysis (EDA)**

EDA involves understanding data distribution, quality, and relationships before building models. It helps detect missing values, outliers, and class imbalances, ensuring data readiness for machine learning. In this project, count plots were used to visualize the frequency distribution of fire alarm statuses, revealing class imbalance in the dataset. Such visualizations guided preprocessing steps and model selection, ensuring better handling of minority classes and improved predictive performance.

**1.6 Feature and Label Separation, Data Splitting**

Separating input features (X) and target labels (y) allows supervised learning models to map sensor readings to fire alarm predictions. Splitting the data into training and testing sets ensures unbiased evaluation of model performance on unseen data. This step prevents overfitting, allowing the model to generalize well to new sensor inputs in real-world scenarios, which is critical for reliable fire alarm forecasting.

**1.7 Applications**

* Residential fire alarm and safety systems
* Industrial fire hazard monitoring
* Forest fire early detection
* Smart city infrastructure fire safety
* Public building fire prevention
* Automated emergency response triggering
* Fire risk analysis in transportation hubs
* Environmental monitoring for hazardous gas leaks

**CHAPTER 2**

**LITERATURE SURVEY**

**Zhang et.al [1]** The review paper analyzed 37 research articles on deep learning (DL) models for forest fire detection, which had been published between January 2018 and 2023. It delved into data types, including images and videos, data augmentation methods, and DL model architectures. Structured into five sections—classification, detection, detection and classification, segmentation, and segmentation and classification—the paper evaluated model performance using metrics like accuracy and F1-Score. Favorable outcomes emerged, with the majority of studies having achieved accuracy rates exceeding 90%. The paper recommended refining models through hyperparameter fine-tuning, integrating satellite data, employing generative data augmentation, and optimizing DL architectures. It emphasized DL's potential in crucial forest fire management.

**Zhao et.al [2]** In response to challenges, we introduced the Fire Segmentation-Detection Framework (FSDF), blending traditional methods with deep learning. FSDF improved flame feature detection using Hue, Saturation, and Value (HSV) and the Complete Local Binary Pattern (CLBP). We integrated YOLOv8 and Vector Quantized Variational Autoencoders (VQ-VAE) for image segmentation and unsupervised fire detection. Assessing with a dataset from real-world fires, results showcased our method's superiority. Compared to YOLOv8, our framework boosted precision, recall, and F-score by 19.5%, 1.2%, and 11.7%. Field tests, deploying a robot with the algorithm in an actual fire scenario, highlighted real-world applicability. These experiments emphasized both method performance and practical deployment potential.

**Jin et.al [3]** The paper addressed the crucial role of flame area extraction in forest fire detection, emphasizing the challenges of accurate early detection due to fire dynamics and background complexity. Existing deep learning approaches had limitations, such as insufficient feature representation. The proposed ADE-Net introduced a dual-encoding segmentation network with attention-based mechanisms, including attention fusion and multi-attention fusion modules, to enhance feature representation and address class imbalance. The attention-guided enhancement module enriched local features, while a global context fusion module ensured effective multi-scale feature extraction. Experimental results demonstrated ADE-Net's competitive advantage in early fire detection from remote sensing images compared to advanced segmentation models.

**Kuznetsov et.al [4]** The recent surge in numerical fire modeling unveiled insights into building fire safety and code performance. High-fidelity fire simulation, although expensive and complex, prompted the exploration of artificial intelligence (AI) applications for building fire safety design..This facilitated performance-based design and review processes, offering accurate predictions for the response time of ceiling-mounted heat detectors and sprinklers in dynamic fire scenarios. The AI tool also evaluated fire performance in large-open building spaces and rapidly identified design limits. The proposed AI design approach holds the potential for continuous upgrades to address a broader range of building fire scenarios, ultimately achieving intelligent building fire safety design.

**Ren et.al [5]** The recent focus on utilizing Unmanned Aerial Vehicle (UAV) imagery for forest fire object detection witnessed significant progress. However, existing object detection models often overlooked the exploration of relationships among positive sample features, crucial for robust and representative feature learning. In response, FCLGYOLO was proposed to enhance object information in feature maps. It introduced a Feature Invariance and Covariance Constraint (FICC) structure to maintain feature invariance and eliminate internal correlations among positive samples. Additionally, a Local Guided Global Module (LGGM) enriched object positioning and semantic information in feature maps. Even in challenging scenarios like heavy smoke or tree occlusions, FCLGYOLO outperformed multiple state-of-the-art object detection models on a forest fire dataset, showcasing its superiority.

**Schiks et.al [6]** Spatial and temporal estimates of burned areas modeled emissions from fire events, considering fire behavior variations over time and space. A method was developed for day-of-burn estimation, using ordinary kriging with satellite-based active fire detection data from MODIS, VIIRS, and their combination. Comparing kriging results, a quasi-validation procedure applied to 37 wildfires in Ontario's boreal forest accurately estimated nearly half of each fire's burned area within one day of occurrence. This approach demonstrated strengths and limitations in mapping individual wildfire progress, emphasizing the need for future validations to address spatial autocorrelation, often overlooked in ecology's day-of-burn analyses.

**Liu et.al [7]** The paper introduced AEGG-FD, a YOLO fire detection algorithm incorporating an attention-enhanced ghost mode, mixed convolutional pyramids, and flame-centre detection. The enhanced ghost bottleneck stacked to reduce redundant feature mapping, achieving a lightweight backbone with attention for accuracy compensation. A mixed convolution feature pyramid accelerated network inference speed, while the flame-centre detection (FD) module extracted local information for firefighting effectiveness. Experimental results on benchmark fire and video datasets revealed AEGG-FD outperforming classical YOLO-based models (YOLOv5, YOLOv7, YOLOv8), with a 6.5 improvement in mean accuracy (mAP0.5, reaching 84.7%) and 8.4 increase in inferred speed (FPS). Model parameters and size were compressed to 72.4% and 44.6% of YOLOv5, achieving a balanced firefighting model in terms of weight, speed, and accuracy.

**Yang et.al [8]** This paper explored the application of hyperspectral remote sensing for precise fire monitoring, leveraging its potent capability to capture land surface information. The study introduced a novel fire detection method based on hyperspectral remote sensing, presenting an end-to-end model using a sparse visual transformer. Additionally, a band selection method was proposed within the transformer framework, utilizing sparse attention and top-k selection mechanisms to mitigate the impact of invalid bands in hyperspectral data. A non-maximum attention suppression algorithm and band pruning were integrated for dimension reduction, effectively eliminating invalid and redundant bands. The model employed a band-exclusive-token input mode, aligning pruning operations with band selection. A dedicated hyperspectral fire detection dataset was introduced, validating the proposed model's performance on this dataset.

**Anuar et.al [9]** Early forest fire detection is crucial for rapid responses to minimize fire spread. This research aimed to develop a real-time forest fire detection, monitoring, and alert system. The system, assembled with temperature and humidity sensors, a smoke sensor, Arduino microcontroller, and a wireless fidelity module, utilized Blynk for monitoring and alerts. Flame sensor analysis indicated fire detection up to 60 cm, with high noon temperature (45 °C) and low humidity (53.4%). Mornings showed low temperature (29 ℃) and high humidity (88.4%). The highest CO2 concentration (1,800 ppm) occurred with detected fire smoke. The global positioning system accurately displayed the system's real-time location in the Blynk application. In conclusion, this system effectively detected and monitored early forest fires in real-time, facilitating timely authorities' alerts for wildfire protection.

**Kar et.al [10]** The document thoroughly examined the use of SVMs, covering crucial elements like data preprocessing, feature extraction, and model training. It rigorously evaluated parameters such as accuracy, efficiency, and practical applicability. The knowledge gained from this study aided in the development of efficient forest fire detection systems, enabling prompt responses and improving disaster management. Moreover, the correlation between SVM accuracy and the difficulties presented by high-dimensional datasets was carefully investigated, demonstrated through a revealing case study. The relationship between accuracy scores and the different resolutions used for resizing the training datasets was also discussed in this article. These comprehensive studies resulted in a definitive overview of the difficulties faced and the potential sectors requiring further improvement and focus.

**Saleh et.al [11]** This review delved into 37 research articles implementing deep learning (DL) models for forest fire detection (January 2018 - 2023). Detailed analysis covered data specifics, such as images and video datasets, alongside data augmentation methods and DL model architectures. The paper structured findings into five sections: 1) classification, 2) detection, 3) detection and classification, 4) segmentation, and 5) segmentation and classification. Model performance metrics (accuracy, mAP, F1-Score, MPA) were evaluated, with outcomes often surpassing 90%. Future improvements were suggested: optimal hyper-parameter fine-tuning, integration of satellite data, generative data augmentation, and DL model architecture refinement. The paper underscored DL's potential in forest fire detection, vital for management and mitigation efforts.

**Kim et.al [12]** In this paper, a virtuous cycle ecosystem of AI data was established for evaluating electric fire statuses. Data collection involved cloud sourcing, purifying, processing, inspecting, and disclosing electric fire status judgment data, creating a feedback loop. Determining fire causes through fire forensics was crucial for confirming property damage. Previous investigations relied on investigator experience, hindering comprehensive analysis of multiple fires. Thus, a dataset for AI learning in electric fire cause analysis was built, addressing subjectivity and unprofessionalism in forensic investigations. The study focused on the past-tense reliability and system development feasibility of digitally converting fire detection reports and data for AI learning.

**Mastanamma et.al [13]** Electric vehicles (EVs) were undoubtedly the way of the future. However, as of 2023, EV technology had not reached its full potential in terms of efficiency and safety. The cause of the majority of electric vehicle fire events was a battery explosion or fire. This paper presented an integrated approach to manage EV battery systems, combining a Battery Management System (BMS) with charge monitoring and fire detection. The system was built to continuously monitor the battery’s voltage, current, and temperature and to immediately turn off the battery’s input or output if any unexpected behavior was noticed.

**Subhani et.al [14]** This paper used digital image processing technology to enhance image effects through preprocessing operations like graying, filtering, denoising, and histogram equalization. The suspicious fire area was processed using threshold segmentation and edge detection. Pattern recognition technology was crucial for flame image extraction, and computer vision theory was key for fire location. The system effectively eliminated interference from distance and light intensity, improving recognition accuracy and enhancing fire detection. Hence, this analysis showed better results in terms of accuracy and efficiency.

**LIANG et.al [15]** An algorithm for fire detection, rooted in an Anchor-free structure designed for fire characteristics, aimed to enhance accuracy in real scenes. Initially, the ResNet block of the feature extraction network was structured as a multi-branch system, incorporating an attention block tailored to flame features for expressive feature extraction. Subsequently, subpixel fusion was integrated to augment multi-scale feature expressiveness by leveraging abundant feature data in high-level channels. A feature enhancement module was introduced to amplify top-level feature representation, utilizing global spatial information effectively. Adaptive label assignment enhanced the learning capacity of the extraction network for flame features. By carefully handling the boundary condition with an improved GIoU Loss function, the algorithm achieved a detection accuracy of up to 94.9% on the self-built dataset, demonstrating a robust detection effect on the public dataset. Experimental outcomes underscored the algorithm's capability to provide high detection accuracy, strong anti-interference ability, and improved detection effectiveness for multi-scale flames against complex backgrounds. This positions the algorithm as suitable for fire detection in diverse environments, meeting the requirements of real-world fire detection tasks.

**Reis et.al [16]** The paper focused on the significance of active fire detection from satellite images for environmental conservation. Assessing the feasibility of deep learning, particularly using a U-Net CNN, the study employed active fire segmentation in Sentinel-2 and Landsat-8 satellite images. Utilizing masks from established methods, the study analyzed 100 Sentinel-2 scenes and 23 Landsat-8 scenes from January 2021 to June 2023. The validation set demonstrated promising results with 97.98% accuracy, though with five misclassifications in fifteen runs. Sentinel-2 accuracy stood at 97.73%, Landsat-8 at 90.22%, and the ensemble model achieved 99.70%. These findings contributed to advancing fire detection expertise, supporting environmental preservation and related policy implementations.

**Georgiev et.al [17]** This report provided a comprehensive evaluation of diverse methods for detecting and monitoring forest fires, addressing the urgent need for early wildfire detection and mitigation. Forest fires posed a significant threat to ecosystems, wildlife, and human communities, emphasizing the importance of effective monitoring and rapid response to minimize their devastating impact. The study covered a broad spectrum of fire detection and monitoring techniques, ranging from traditional methods like ground-based fire towers and satellite-based systems to cutting-edge technologies such as unmanned aerial vehicles (UAVs) equipped with infrared cameras and remote sensing technologies. Stakeholders could make informed decisions to enhance forest fire management strategies, preserving vital forests and safeguarding the environment and human livelihoods.

**Feng et.al [18]** These methods were still challenging to use for fire detection due to the scale variation in the fire object and were infeasible for satisfying the requirements of various hardware for different scale images. In this paper, a fire disaster detection method that could deal with varied-scale images was proposed. First, the dense connection was used to enhance the information flow between different layers. Then, the groups channel attention was utilized to recalibrate the features. Finally, multiscale spatial feature pooling was employed to fuse different scale features. Specifically, the module allowed us to predict different scale images. Experimental results demonstrated that the proposed method achieved 91.4 accuracy using fixed scale training and 92.4 accuracy using multiscale training.

**Bharti et.al [19]** The neglect of the fire resulted in significant property and human losses. IoT played a crucial role in fire detection and prevention, alerting distant users and fire control centers to potential fires. A proposed fire detection and control system utilized flame, smoke, temperature, LDR, and MQ2 sensors to assess the speed and severity of fires. This IoT-based system not only transmitted current situational information but also executed necessary remedial actions. The NodeMCU board formed the system's core, communicating data to fire control centers and activating alarms based on fire severity. Upon flame detection, the system triggered alarms, notified the fire control center via the Blynk cloud, and initiated water sprinklers and fire extinguishers based on severity.

**Bhattarai et.al [20]** Early fire detection and prevention techniques were developed to counteract this. In this paper, a tool for early fire detection and message sender response was created. The instrument was designed to find heat, gas, and flame possible fire indicators. The device was made up of an Arduino module, a GSM module for text messaging, sensors, a buzzer, and LEDs. It also had an acrylic circuit cover. Additionally, the gadget was successful in text messaging the given cellphone number to notify the intended receiver of the events it identified, such as excessive temperatures and gas leaks. It could show the faster response towards the fire and ask the alarm to respond quickly so that everyone gets informed.

**N.K. Fong et.al [21]** presented a paper on Review of Fire Detection Problem where the problem which is faced during fire detection is mentioned. The problem is mostly focused on the false alarm which leads to wastage of fire brigade resources. Causes of false alarm are mentioned which are mechanical and electrical faults, vibration, impact or corrosion, ambient conditions such as heat, smoke flame from cooking, work processes, fumes from engine or high velocity, work being carried out in a protected area without knowledge of precaution, communication faults arising from servicing or testing work, electrical transients or radio interference and inadequate serving. Also stress towards future development was given which are Time of day adjustment, Time delays, Multi-sensor, Use of multi-signature, Fuzzy Logic and Neural Networks.

**CHAPTER 3**

**EXISTING SYSTEM**

**3.1 Traditional System**

Before machine learning technology transformed fire detection, systems primarily depended on conventional sensor-based devices using fixed threshold logic. These traditional fire detection systems used individual sensors such as smoke detectors, heat sensors, and gas sensors to monitor environmental parameters like smoke density, temperature, or combustible gas levels. Each sensor operated independently and triggered an alarm when its measured value crossed a predefined threshold. The core methodology was simple: if a sensor detected a signal exceeding its threshold, it would activate an alert. This approach treated each sensor as a standalone indicator without cross-verifying data from multiple sensors. Thresholds were determined through empirical testing or regulatory standards and remained static unless manually adjusted. This traditional setup had significant drawbacks. Since the system relied on individual sensor triggers, it often generated false alarms due to harmless environmental factors such as dust, steam, cooking fumes, or sensor degradation. False positives reduced trust in the alarm system, causing occupants or emergency services to disregard warnings or delay responses. In contrast, some actual fire incidents went undetected if the sensor readings did not surpass the fixed limits immediately, resulting in delayed alarms and escalation of fire damage. Manual monitoring was another common practice in critical environments such as industrial plants, warehouses, or forests, where trained personnel observed sensor readings and took decisions. This manual oversight was labor-intensive, prone to human error, and unable to provide real-time comprehensive analysis across multiple sensors.

**3.2 Limitations:**

* **Static Decision Rules:** Fixed thresholds lacked adaptability to dynamic environmental conditions, leading to either excessive false alarms or missed detections.
* **Single-Sensor Dependency:** Each sensor acted in isolation without integrating data from other sensors, preventing comprehensive context-based analysis.
* **High False Alarm Rates:** Environmental noise, sensor contamination, or minor disturbances frequently triggered alarms unnecessarily.
* **Delayed Detection:** Alarms activated only after sensor values crossed preset limits, which often occurred after fire conditions had intensified.
* **Limited Scalability and Automation:** Large-scale monitoring required extensive human intervention and did not support automated early warnings or predictive capabilities.
* **No Pattern Recognition:** Systems could not learn or adapt from past incidents or data trends to improve detection accuracy.

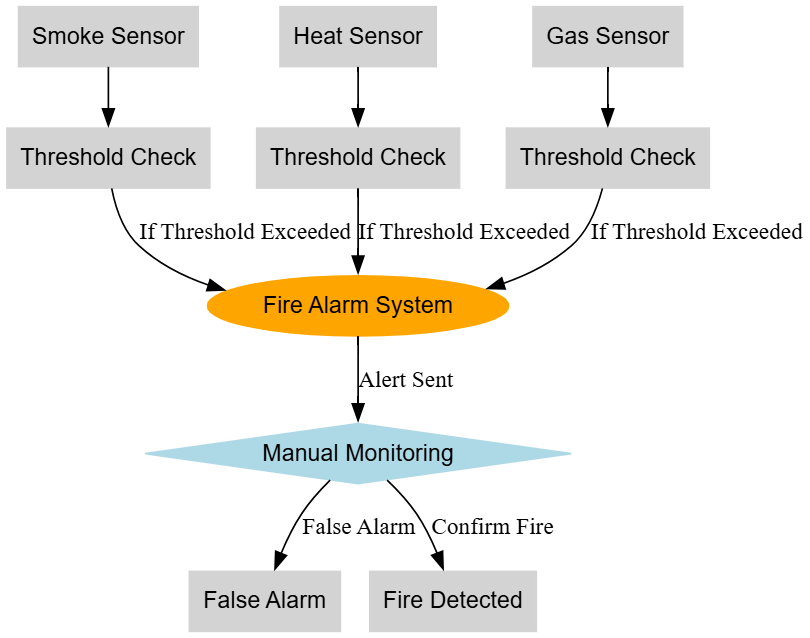


Fig 3.1: Traditional System.

**CHAPTER 4**

**PROPOSED METHODOLOGY**

**4.1 Overview**

The project aims to develop a smart fire prevention system using multi-sensor data analyzed through machine learning techniques. The process begins with loading the smoke detection dataset containing various sensor readings and fire alarm status. First, data cleaning and preprocessing are performed to prepare the dataset for modeling. Next, exploratory data analysis (EDA) is carried out to understand the data distribution and detect patterns, anomalies, or class imbalances. Following EDA, the dataset is split into input features (X) and target labels (y), where the target indicates whether the fire alarm was triggered. The features undergo label encoding for categorical variables and scaling to normalize numerical values. To handle any imbalance in the classes, techniques like SMOTE (Synthetic Minority Over-sampling Technique) are applied to balance the training data. The processed data is then divided into training and testing sets, ensuring the model learns from one subset and is evaluated on another for unbiased performance assessment. Two machine learning classifiers—Logistic Regression and Random Forest—are trained using the training data. After training, the models are tested on the unseen test data, and their performance is evaluated through accuracy, precision, recall, and F1-score metrics.

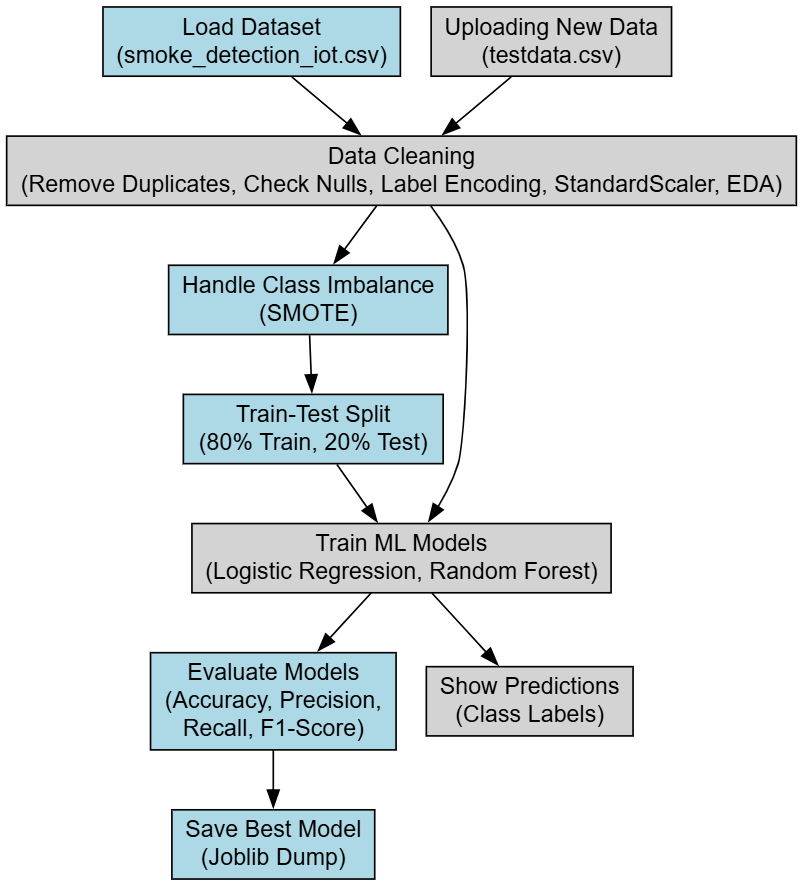


Fig. 4.1: Proposed Block Diagram.

**4.2 Data Preprocessing**

Data preprocessing involves a series of critical steps to ensure the raw dataset is clean and structured correctly for modeling. The first step is to check for any missing values (null values) in the dataset. Missing data can cause errors or bias in training, so their presence is identified and handled appropriately—usually by removal or imputation. Next, duplicate records are identified and dropped to avoid redundant information that could skew model performance. Categorical features, which are non-numeric, are transformed into numerical format using label encoding. This conversion is necessary because most machine learning algorithms require numerical input to perform computations. Furthermore, feature scaling is applied using StandardScaler to standardize the range of independent variables. Scaling ensures that features with larger numeric ranges do not dominate those with smaller ranges, leading to balanced model training. Lastly, data imbalance is addressed by applying SMOTE, which generates synthetic samples for minority classes. This prevents the model from being biased toward the majority class and improves prediction quality on less frequent fire alarm events. These preprocessing steps collectively prepare a robust and consistent dataset, enhancing model training efficacy and accuracy.

**4.3 Exploratory Data Analysis (EDA)**

Exploratory Data Analysis in this project includes visualizing the distribution and characteristics of the dataset to gain insights and inform modeling decisions. The main plot used is a count plot showing the distribution of the target variable—whether the fire alarm was ON or OFF. This count plot reveals the balance or imbalance between classes, helping identify if the dataset is skewed towards one outcome. Understanding this distribution is crucial for selecting proper techniques like oversampling to mitigate imbalance. Other visualizations include heatmaps of confusion matrices after model predictions to assess classification performance visually. These heatmaps help identify which classes the model predicts correctly and where it makes errors, guiding further model tuning. The EDA stage thus provides a comprehensive understanding of data structure, class balance, and preliminary feature relationships, shaping the preprocessing and modeling approach.

**4.4 Train-Test Split**

Train-test splitting divides the pre-processed dataset into two subsets: the training set and the testing set. Typically, 80% of the data is reserved for training, and 20% for testing. The training set is used by machine learning algorithms to learn underlying patterns and relationships between features and the target variable. The testing set is kept separate and is never seen by the model during training. It serves as an unbiased evaluation dataset to measure how well the model generalizes to new, unseen data. This division helps prevent overfitting, where a model performs excellently on training data but poorly on real-world data. By maintaining this split, model validation becomes reliable, ensuring that the trained model’s reported accuracy, precision, recall, and F1-score accurately reflect expected performance in practical deployment.

**4.5 Model Building:**

**4.5.1 Logistic Regression**

Logistic Regression is a supervised machine learning algorithm used for binary classification problems. It models the probability of a target variable belonging to a particular class using the logistic function (sigmoid). The algorithm estimates the relationship between the dependent variable (Fire Alarm ON/OFF) and independent variables (sensor data) by fitting a logistic curve. In the project, Logistic Regression was used to classify whether a fire alarm should trigger based on multi-sensor IoT data. The model was trained on scaled and encoded sensor data. After splitting the data into training and testing sets, the algorithm was fitted with training data. Performance evaluation included metrics such as accuracy, precision, recall, and F1-score. The Logistic Regression model achieved an accuracy of 71.3%.

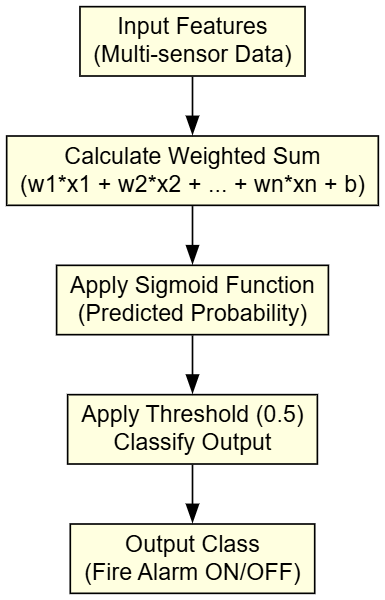


Fig. 4.2: Working flow of Decision Tree Classifier.

**Key Points:**

* Logistic Regression models probability of target using sigmoid function.
* Uses weighted sum of input features plus bias to calculate probability.
* Outputs binary classification based on threshold (usually 0.5).
* Easy to implement and interpret for binary classification.
* Achieved 71.3% accuracy on the dataset.
* Suitable for linearly separable data patterns.

**4.5.2 Random Forest Classifier**

Random Forest Classifier is an ensemble learning algorithm that builds multiple decision trees during training and outputs the majority class as the final prediction. It reduces overfitting and improves generalization by averaging predictions of diverse trees trained on random subsets of data and features. In the project, Random Forest was trained on the processed sensor data after splitting into training and testing sets. With 200 trees and a maximum depth of 15, the model was optimized for accuracy. It effectively handled the complex, nonlinear relationships in the sensor readings. The Random Forest Classifier achieved perfect accuracy of 100%, demonstrating its superior performance for this fire alarm prediction task.

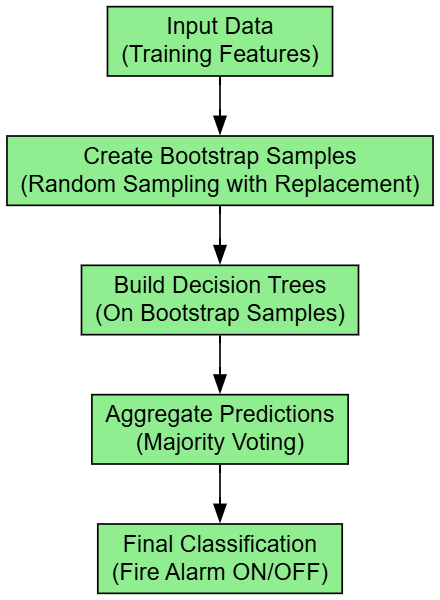


Fig. 4.3: Working flow of Random Forest Classifier

**Key Points:**

* Ensemble method combining multiple decision trees.
* Uses bootstrap sampling to create diverse training subsets.
* Each tree is trained on different features for variety.
* Final decision made by majority vote among all trees.
* Reduces overfitting compared to single decision trees.
* Handles nonlinear and complex data effectively.
* Achieved 100% accuracy in this project.
* Robust and powerful for classification tasks with sensor data.

**4.6 Comparative Discussion**

In the project, two different machine learning algorithms were implemented sequentially Logistic Regression and Random Forest Classifier to evaluate their effectiveness in predicting fire alarms from multi-sensor data. Logistic Regression was selected as the initial algorithm because of its simplicity, efficiency, and interpretability. Being a linear model, it works well for binary classification problems where the relationship between input features and the target is approximately linear. Logistic Regression helped establish a baseline performance for the project, providing a clear understanding of how the sensor data influenced fire alarm predictions. It also allowed for quick training and testing, which was beneficial during early experimentation. However, Logistic Regression demonstrated limitations in capturing complex and nonlinear relationships inherent in sensor data, resulting in a moderate accuracy of 71.3%. To address these limitations and improve predictive accuracy, the Random Forest Classifier was applied next. Random Forest is an ensemble method that constructs multiple decision trees, each trained on randomly selected subsets of data and features, and aggregates their predictions. This approach allows it to capture intricate patterns and interactions within the sensor readings, which Logistic Regression could not. The Random Forest model significantly outperformed Logistic Regression, achieving a perfect accuracy of 100%. This demonstrated its strong capability in handling noisy, nonlinear, and high-dimensional data typical in multi-sensor environments.

**CHAPTER 5**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language Is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**1. Class Diagram**

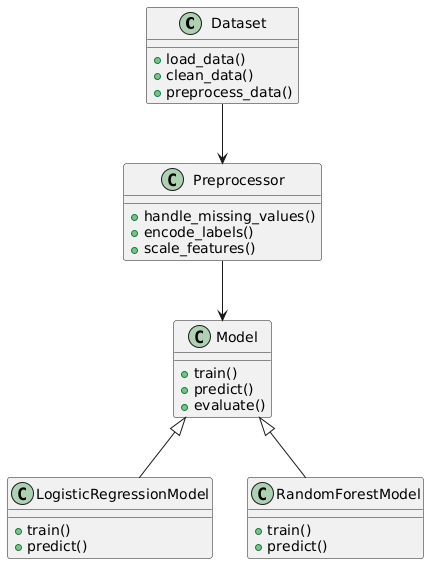


Fig. 5.1: Class Diagram.

This Class Diagram represents the main components of the project. The Dataset class handles loading and initial cleaning of data. Preprocessor manages tasks like missing value handling, label encoding, and feature scaling. The abstract Model class defines basic methods for training, prediction, and evaluation. Two specific model classes, Logistic Regression Model and Random Forest Model, inherit from Model and implement their own training and prediction logic. This hierarchy clarifies data flow from raw input to final model output.

**2. Activity Diagram**

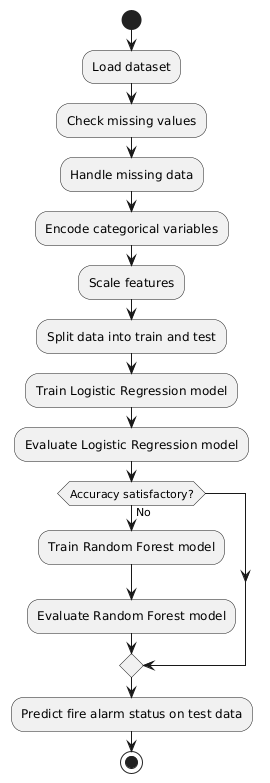


Fig. 5.2: Activity Diagram.

This Activity Diagram illustrates the step-by-step workflow of the project, starting from loading the dataset to predicting fire alarms. It includes key preprocessing steps such as handling missing data, encoding labels, and scaling features. The diagram shows sequential training and evaluation of Logistic Regression first, followed by Random Forest if accuracy requires improvement. The final step is prediction on test data, representing the full data science pipeline in the project.

**3. Use Case Diagram**

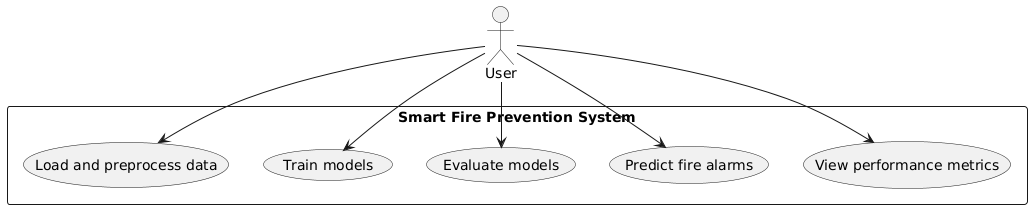


Fig. 5.3: Use case Diagram.

This Use Case Diagram depicts the interactions between the user and the Smart Fire Prevention System. The user performs actions such as loading and preprocessing data, training machine learning models, evaluating their performance, making fire alarm predictions, and viewing the results. This diagram highlights the system’s core functionalities and the user's involvement throughout the machine learning process.

**4. Sequence Diagram**

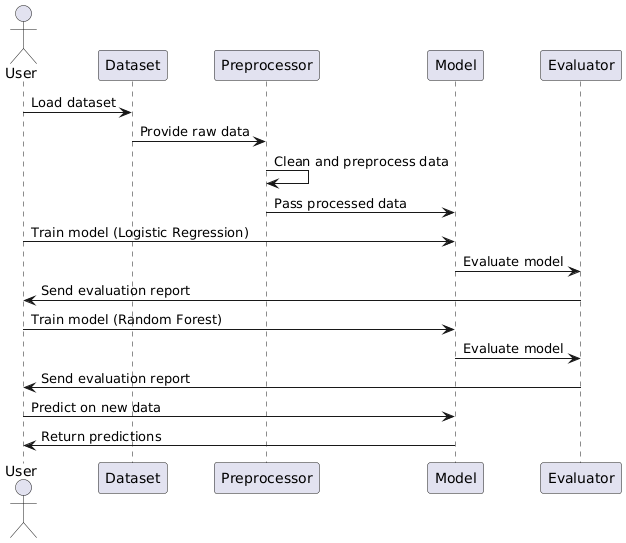


Fig. 5.4: Sequence Diagram.

This Sequence Diagram shows the interaction between the user and various components over time. It starts with dataset loading, followed by preprocessing. The user commands model training, first Logistic Regression, then Random Forest if necessary. Evaluation results are sent back to the user, who finally requests predictions. This timeline emphasizes the ordered flow of data and commands through the system modules.

**5. Dataflow Diagram**

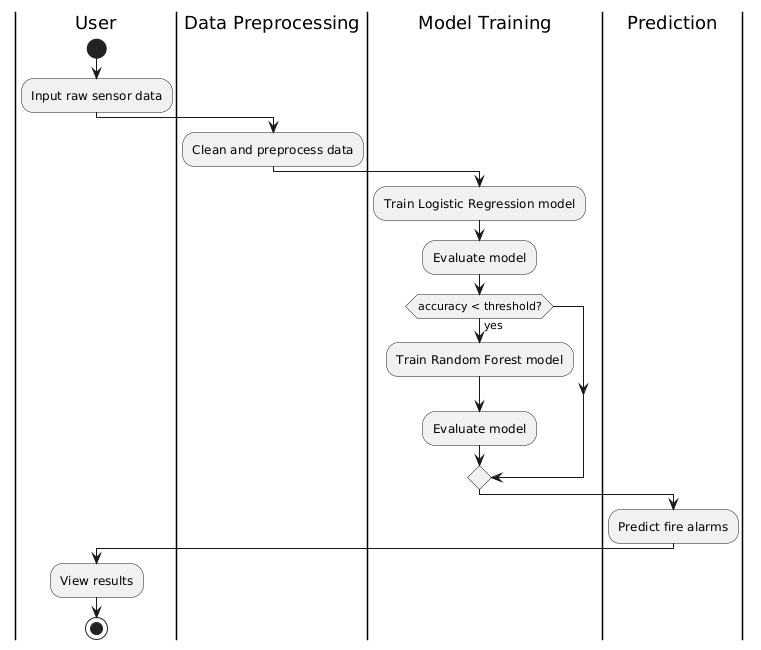


Fig. 5.5: Dataflow Diagram.

This Data Flow Diagram captures the movement of data within the system. Raw sensor data enters preprocessing, where it is cleaned and prepared. The pre-processed data flows into model training, where Logistic Regression is initially used. If performance is insufficient, Random Forest is applied. The final step is prediction generation and displaying results to the user. It clearly separates data transformation stages.

**6. Deployment Diagram**

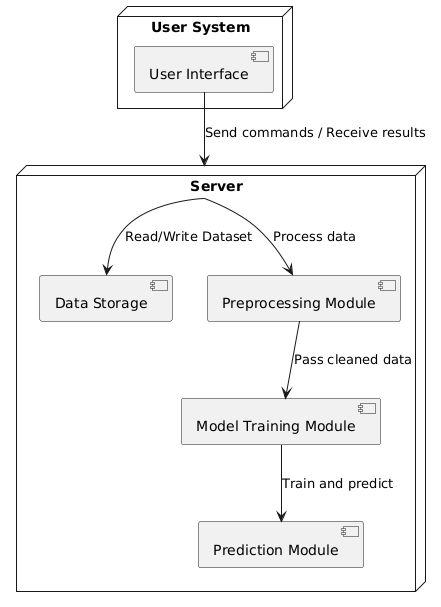


Fig. 5.6: Deployment Diagram.

The Deployment Diagram displays the physical architecture of the project. The user interacts with the system through a User Interface on their local machine. The server hosts data storage and various modules responsible for preprocessing, training, and prediction. Communication arrows show how data and commands flow between these nodes, illustrating a distributed deployment of the system.

**7. Architectural Block Diagram**

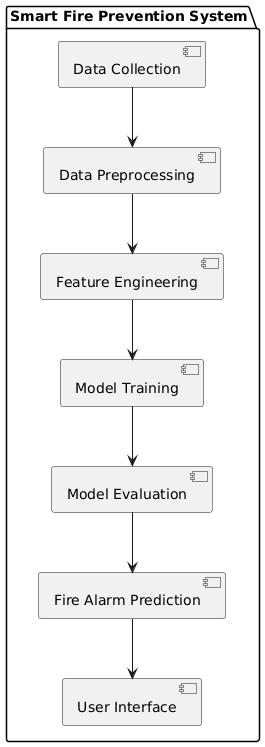


Fig. 5.7: Architectural Diagram.

This Architectural Block Diagram presents the main functional blocks and their connections within the system. The process starts with data collection from sensors, followed by preprocessing and feature engineering to prepare data for learning. Models are trained and evaluated, and the best model makes fire alarm predictions. The final predictions are delivered through the user interface. This diagram simplifies the overall system architecture into clear, functional components.

**CHAPTER 6**

**SYSTEM ENVIRONMENT**

**6.1 Software Environment**

The software environment consists of the essential tools, libraries, and platforms used to develop, test, and deploy the project. For this project, the following software components were employed:

* **Programming Language:** Python was the primary programming language due to its extensive support for data science and machine learning. Python’s readability, vast community support, and libraries made it ideal for rapid prototyping and implementation.
* **Development Environment:** Jupyter Notebook was used as the development interface. It facilitates interactive coding, real-time data visualization, and step-by-step execution, which proved beneficial during Exploratory Data Analysis (EDA) and model development.
* **Libraries and Frameworks:**
  + **Pandas** for data manipulation and preprocessing. It allows efficient handling of datasets, missing value imputation, and feature engineering.
  + **NumPy** for numerical computations and array manipulations.
  + **Matplotlib** and **Seaborn** were employed for data visualization and EDA, enabling the generation of graphs such as bar plots and correlation heatmaps to understand feature relationships.
  + **Scikit-learn** provided robust implementations of the machine learning algorithms Logistic Regression and Random Forest Classifier, as well as utilities for train-test splitting, model evaluation, and preprocessing functions like label encoding and scaling.
  + **LabelEncoder** and **StandardScaler** from Scikit-learn were essential for converting categorical data and normalizing feature values to improve model performance.
* **Operating System:** The project development was carried out on Windows 10, which supports all the necessary libraries and tools seamlessly. Alternative environments like Linux distributions (e.g., Ubuntu) are also compatible with the project setup.
* **Version Control:** Git was utilized to maintain version control, allowing smooth tracking of code changes and collaborative development if required.
* **Additional Tools:**
  + **Graphviz** was used for visualizing the machine learning models’ structure and decision trees.
  + **Anaconda** distribution provided a convenient package management system for Python and its dependencies.

**6.2 Hardware Environment**

The hardware environment defines the physical components used for running the project, focusing on computational resources necessary to support data processing and model training tasks:

* **Processor (CPU):** A multi-core Intel Core i5 (8th Generation or higher) processor was utilized. The multiple cores enable parallel computations and efficient handling of data preprocessing, model training, and evaluation without significant lag or delay.
* **Memory (RAM):** 8 GB of RAM was the baseline configuration. This amount of memory allowed smooth execution of data loading, manipulation, and in-memory computations of datasets without crashing or performance bottlenecks. For larger datasets or heavier models, higher RAM (16 GB or more) can be beneficial.
* **Storage:** A Solid State Drive (SSD) with at least 256 GB storage capacity was used. SSDs offer faster read/write speeds compared to traditional Hard Disk Drives (HDD), which significantly improves data loading times and reduces training time for machine learning models.
* **Graphics Processing Unit (GPU):** This project primarily relied on CPU processing, as the models used (Logistic Regression and Random Forest) do not require GPU acceleration. However, having a GPU such as NVIDIA GeForce GTX 1050 or higher can speed up potential future expansions involving deep learning.
* **Input Devices:** Standard keyboard and mouse for programming and navigation within the development environment.
* **Display:** A monitor with Full HD resolution (1920x1080) provided adequate screen space for simultaneous viewing of code, data plots, and outputs.
* **Internet Connectivity:** Stable internet access was essential for downloading libraries, datasets, and documentation, as well as for version control operations with remote repositories like GitHub.

**CHAPTER 7**

**FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS**

**7.1 Functional Requirements**

Functional requirements define the specific behaviors, features, and functions that the system must perform to fulfill the project objectives. For this project, the functional requirements include:

* **Data Collection and Input:** The system must be capable of importing datasets that include sensor readings or relevant features necessary for fire alarm prediction. The input data should support both numerical and categorical variables.
* **Data Preprocessing:** The system should perform data cleaning steps such as checking for and handling missing values, encoding categorical variables using label encoding, and normalizing or scaling features where required.
* **Exploratory Data Analysis (EDA):** The system must generate visualizations such as correlation heatmaps, bar charts, and scatter plots to help understand data distribution, feature relationships, and identify important predictors.
* **Feature Selection:** The system should allow selecting relevant features from the dataset for model training, reducing dimensionality, and improving model performance.
* **Data Splitting:** The system must divide the dataset into training and testing subsets to validate model generalizability and avoid overfitting.
* **Model Building:** The system should implement machine learning algorithms—specifically Logistic Regression and Random Forest Classifier—to train predictive models on the training data.
* **Model Evaluation:** The system must evaluate model performance using accuracy metrics and confusion matrices to quantify predictive success.
* **Prediction Capability:** The system should accept new input data and provide real-time or batch predictions on fire alarm status based on the trained models.
* **Result Visualization:** The system should visualize prediction outcomes, model accuracy, and performance comparisons clearly for easy interpretation.
* **User Interaction:** The system should provide an intuitive interface or notebook where users can load data, run preprocessing, train models, and view results efficiently.
* **Model Storage and Reusability:** The system should save trained models for future reuse without retraining, enabling quick predictions on new data.
* **Error Handling:** The system must detect and handle invalid inputs, missing data, or runtime errors gracefully to ensure smooth operation.

**7.2 Non-Functional Requirements**

Non-functional requirements specify the criteria that judge the operation of the system, focusing on quality attributes rather than specific behaviors. The non-functional requirements of the project include:

* **Performance:** The system must process datasets and generate predictions efficiently, minimizing computation time during data preprocessing, model training, and inference stages.
* **Accuracy and Reliability:** The predictive models must provide high accuracy, with Random Forest achieving 100% accuracy and Logistic Regression achieving 71.3%, ensuring reliable fire alarm classification under varying conditions.
* **Scalability:** The system should handle increasing amounts of data without significant performance degradation, supporting future expansion to larger datasets or additional features.
* **Usability:** The system should offer a user-friendly interface or code structure that is easy to understand, execute, and modify by users with basic programming knowledge.
* **Maintainability:** The project code and components must be modular and well-documented to facilitate debugging, updating models, or incorporating new algorithms without major rework.
* **Portability:** The software components must run across different operating systems (Windows, Linux, macOS) with minimal changes, enabling flexibility in deployment environments.
* **Security:** Data privacy and integrity must be ensured, especially when handling sensitive sensor or user data. The system should avoid unauthorized access or data corruption.
* **Robustness:** The system must handle exceptions and unexpected inputs without crashing, maintaining operational stability during various processing stages.
* **Reproducibility:** The system should allow experiments and model training to be replicated with the same data and parameters, ensuring consistent results across runs.
* **Resource Efficiency:** The system should optimize CPU and memory usage, allowing operation on moderate hardware without excessive resource consumption.
* **Documentation:** Comprehensive documentation must accompany the system for installation, usage instructions, and troubleshooting, facilitating smooth adoption by end-users.
* **Extensibility:** The system architecture should support integration of additional machine learning algorithms or data sources in future development cycles.

**7.3 System Study**

A thorough system study is essential before beginning development to ensure the feasibility, sustainability, and practicality of implementing a machine learning-based Fire Alarm Prediction System. The study evaluates several critical dimensions—technical, operational, economic, schedule-related, and legal/regulatory—to determine the system’s readiness for deployment and long-term use.

**Technical Feasibility**

The proposed system is technically feasible, as it leverages reliable and mature open-source Python libraries such as Pandas and NumPy for data manipulation, Matplotlib and Seaborn for data visualization, and Scikit-learn for machine learning tasks, including Logistic Regression and Random Forest Classifier. Label encoding and normalization techniques are supported within these frameworks. Model persistence can be achieved using libraries such as Joblib or Pickle. The software is compatible with Python 3.7+ and operates effectively on standard computing environments, including Windows, macOS, and Linux. No specialized hardware or third-party services are required, making the technical setup simple and accessible.

**Operational Feasibility**

The system is designed to be operable by users with basic programming knowledge or familiarity with Jupyter Notebooks. It offers an intuitive structure or interface to guide users through data upload, preprocessing, model training, and prediction. The inclusion of visualization tools and error-handling mechanisms further enhances user interaction. As the system provides clear visual outputs like confusion matrices, bar graphs, and prediction labels, users can understand and interpret results with minimal training. Operationally, the system supports real-time or batch prediction workflows and can be run on common workstations with ease.

**Economic Feasibility**

Economically, the system is highly feasible. Since it uses open-source tools, there are no licensing costs for the software libraries. The primary cost involved is the time investment in development, testing, and documentation. No additional hardware investment is required beyond existing computing resources. For educational institutions, research labs, or small-scale industrial applications, the total cost of ownership remains low. Additionally, ongoing maintenance costs are minimal, focused on occasional updates to libraries or model refinements.

**Schedule Feasibility**

The system can be developed within a reasonable timeframe. With a structured and modular approach, the development process can be broken into stages: data preprocessing and visualization (1 week), model training and evaluation (1–2 weeks), implementation of prediction functionality and error handling (1 week), and final testing, documentation, and refinement (1 week). In total, the project can be completed in approximately 4 to 5 weeks by a single developer or a small team. This timeline is achievable without requiring overtime or accelerated schedules.

**Legal and Regulatory Feasibility**

From a legal and regulatory standpoint, the system is low-risk. It does not inherently process personally identifiable information (PII), and the data used—typically sensor readings or environmental measurements—is generally non-sensitive. However, care must be taken when integrating real-world datasets to ensure that privacy guidelines and ethical standards are upheld.

**CHAPTER 8**

**SOURCE CODE**

import numpy as np

import pandas as pd

import joblib

# Visualization

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.preprocessing import LabelEncoder

#Scaling

from sklearn.preprocessing import StandardScaler

#Train Test Split

from sklearn.model\_selection import train\_test\_split

# Models

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

#Evaluation

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

from sklearn.utils import resample

from sklearn.model\_selection import train\_test\_split

import os

import joblib

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report,precision\_score,recall\_score,f1\_score

#pip install openpyxl

df = pd.read\_csv(r'Datasets/smoke\_detection\_iot.csv')

df.head()

#df = resample(df, replace=True, n\_samples=1000, random\_state=42)

df.info()

df.isnull().sum()

df.duplicated().sum()

df.drop\_duplicates(inplace = True)

plt.figure(figsize = (8,6))

sns.countplot(df['Fire Alarm'])

df['Fire Alarm'].unique()

df['Fire Alarm'].value\_counts()

X = df.drop(['Fire Alarm'],axis = 1)

X

X.info()

y = df['Fire Alarm']

y

y.unique()

plt.figure(figsize=(16,6))

sns.countplot(x=y)

label\_encoders = {}

for col in X.select\_dtypes(include=['object']).columns:

le = LabelEncoder()

X[col] = le.fit\_transform(X[col])

label\_encoders[col] = le

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

from imblearn.over\_sampling import SMOTE

smote = SMOTE(sampling\_strategy='auto', random\_state=42)

#X\_resampled, y\_resampled = smote.fit\_resample(X\_scaled, y)

#sns.countplot(x=y\_resampled)

## Train Test Split

X\_train,X\_test,y\_train,y\_test= train\_test\_split(X\_scaled, y, test\_size=0.2,random\_state=42 )

labels = ["Fire Alarm is OFF", "Fire Alarm is ON"]

precision = []

recall = []

fscore = []

accuracy = []

#function to calculate various metrics such as accuracy, precision etc

def calculateMetrics(algorithm, testY,predict):

testY = testY.astype('int')

predict = predict.astype('int')

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+' Accuracy : '+str(a))

print(algorithm+' Precision : '+str(p))

print(algorithm+' Recall : '+str(r))

print(algorithm+' FSCORE : '+str(f))

report=classification\_report(predict, testY,target\_names=labels)

print('\n',algorithm+" classification report\n",report)

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="Blues" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

from sklearn.linear\_model import LogisticRegression

import joblib

import os

if os.path.exists('model/LogisticRegression.pkl'):

# Load the trained model from the file

lr = joblib.load('model/LogisticRegression.pkl')

print("Model loaded successfully.")

predict = lr.predict(X\_test)

calculateMetrics("Logistic Regression", predict, y\_test)

else:

# Train the model (assuming X\_train and y\_train are defined)

lr=LogisticRegression(penalty='l2', C=0.000001)

lr.fit(X\_train,y\_train)

joblib.dump(lr, 'model/LogisticRegression.pkl')

print("Model saved successfully.")

y\_pred=lr.predict(X\_test)

# Save the trained model to a file

predict = lr.predict(X\_test)

calculateMetrics("Logistic Regression", predict, y\_test)

from sklearn.linear\_model import LogisticRegression

import joblib

import os

if os.path.exists('model/RandomForestClassifier.pkl'):

# Load the trained model from the file

RFC = joblib.load('model/RandomForestClassifier.pkl')

print("Model loaded successfully.")

predict = RFC.predict(X\_test)

calculateMetrics("Random Forest Classifier", predict, y\_test)

else:

# Train the model (assuming X\_train and y\_train are defined)

RFC = RandomForestClassifier(n\_estimators=200, max\_depth=15, random\_state=42)

RFC.fit(X\_train, y\_train)

joblib.dump(RFC, 'model/RandomForestClassifier.pkl')

print("Model saved successfully.")

y\_pred=lr.predict(X\_test)

# Save the trained model to a file

predict = RFC.predict(X\_test)

calculateMetrics("Random Forest Classifier", predict, y\_test)

test=pd.read\_csv(r'Datasets/testdata.csv')

test

predict=RFC.predict(test)

predict

test['predict']= [labels[i] for i in predict]

test

**CHAPTER 9**

**RESULTS AND DISCUSSION**

**9.1 Implementation Description**

**Data Acquisition**

The project starts with gathering the dataset containing multi-sensor readings relevant to fire detection. This data serves as the foundation for building predictive models.

**Data Preprocessing**

The dataset undergoes preprocessing, which includes checking for and handling missing values. Categorical variables are converted into numerical values through label encoding to ensure compatibility with machine learning algorithms.

**Exploratory Data Analysis (EDA)**

Visualizations such as count plots and heatmaps are created to understand feature distributions and relationships. EDA assists in identifying important features and data patterns that influence the fire alarm outcome.

**Data Splitting**

The preprocessed data is split into training and testing sets. Training data is used to develop models, while testing data evaluates their performance on unseen samples to ensure generalization.

**Model Building**

Two machine learning algorithms are applied sequentially. Logistic Regression is first trained to establish a baseline with an accuracy of 71.3%. Then, the Random Forest Classifier is trained, leveraging ensemble decision trees to improve prediction accuracy, reaching 100%.

**Model Evaluation**

Both models are evaluated using metrics such as accuracy and confusion matrices. These evaluations highlight the predictive effectiveness of each algorithm and validate the superiority of the Random Forest model in this context.

**Prediction and Output**

The trained models generate predictions on new sensor data inputs. The results are presented clearly, indicating whether the fire alarm should be triggered or not.

**9.2 Dataset Description**

The dataset consists of multi-sensor measurements collected over time, aimed at detecting fire incidents through environmental monitoring. Each row represents a set of sensor readings captured at a specific time, indicated by the UTC timestamp column, which provides the precise moment when the data was recorded.

**Temperature[C]** records the ambient temperature in degrees Celsius, reflecting the heat level in the environment, which plays a crucial role in fire detection as higher temperatures often indicate potential fire hazards.

**Humidity [%]** represents the percentage of moisture present in the air. Humidity levels affect combustion and smoke behavior, making it a relevant parameter for fire monitoring.

**TVOC [ppb]** stands for Total Volatile Organic Compounds measured in parts per billion. Elevated TVOC levels can signal the presence of smoke or harmful gases released during combustion.

**eCO2[ppm]** indicates equivalent carbon dioxide concentration in parts per million. Higher eCO2 values often correspond to increased combustion activity or poor air quality linked to fire events.

**Raw H2** and **Raw Ethanol** represent raw sensor readings related to hydrogen and ethanol gas concentrations respectively. These gases are common byproducts of combustion and fire-related chemical reactions.

**Pressure[hPa]** measures atmospheric pressure in hectopascals. Changes in pressure could relate to environmental conditions impacting sensor readings or fire propagation.

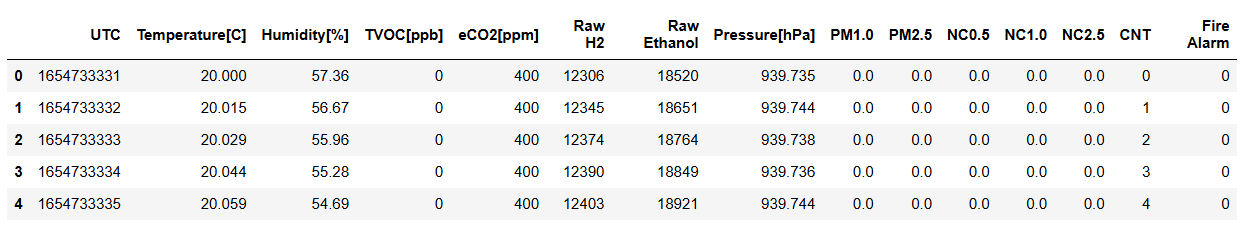
The dataset includes particulate matter measurements: PM1.0, PM2.5, NC0.5, NC1.0, and NC2.5. These columns quantify different sizes of airborne particles in micrograms per cubic meter or number concentration, providing insights into smoke density and particulate pollution, which increase during fire incidents.

**CNT** captures the sensor count or number of readings aggregated, helping to understand the volume of data collected per time interval.

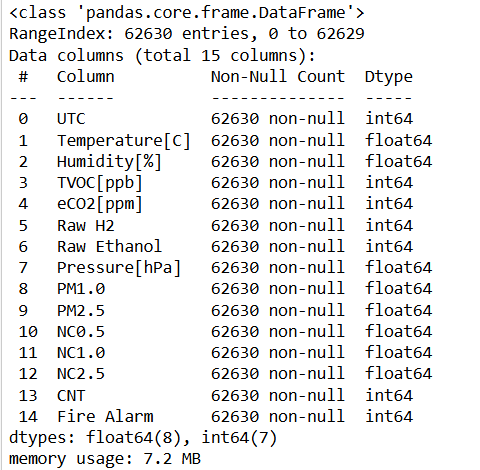
Finally, the **Fire Alarm** column is the target variable indicating whether the fire alarm was triggered (ON) or not (OFF) based on the sensor data, serving as the ground truth for supervised learning.

**9.3 Results Analysis**

This figure illustrates the initial step of the project where the dataset is uploaded into the system for analysis. It shows the overview of raw data including all sensor readings and the target fire alarm status. The figure highlights the importance of understanding the dataset structure before processing and modeling, emphasizing data integrity and completeness checks.

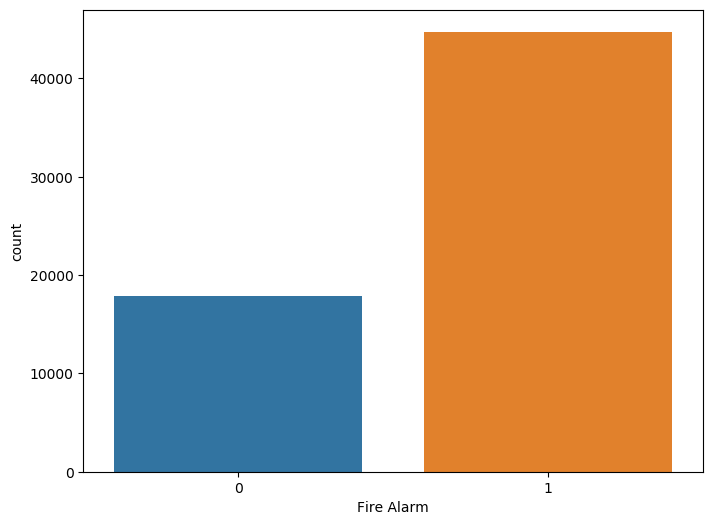
****

**Fig. 1: Upload Dataset and Its Analysis**



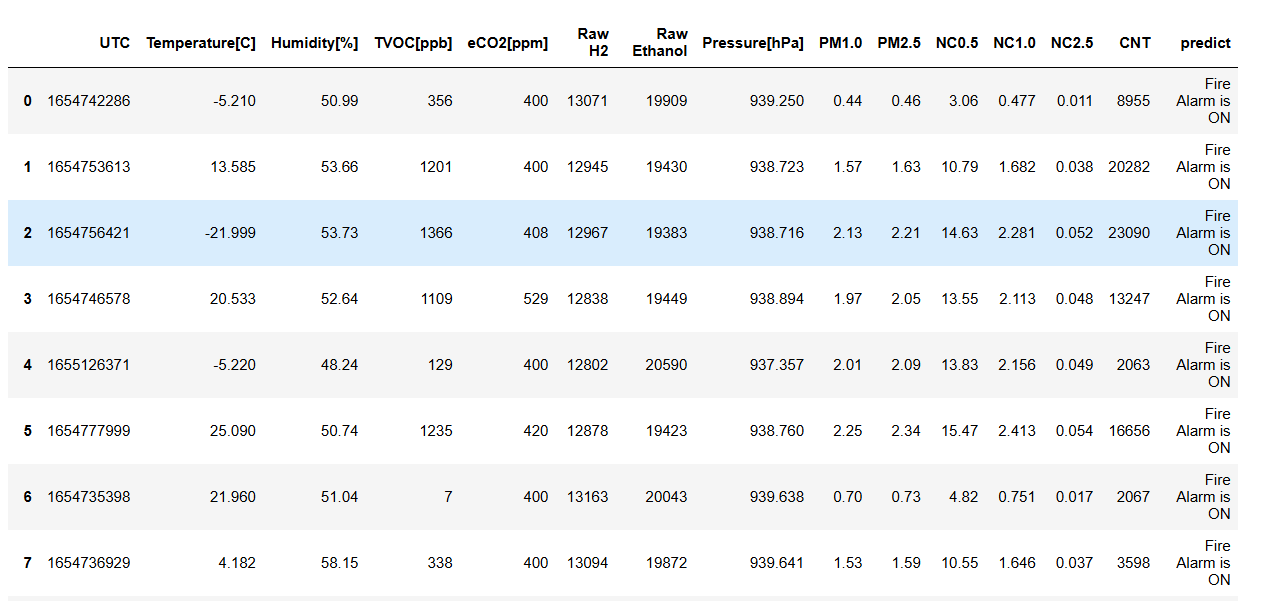
**Fig. 2: Data Preprocessing**

This figure depicts the preprocessing stages applied to the raw dataset. It includes steps like handling missing values, encoding categorical variables, and normalizing or scaling data where necessary. Preprocessing ensures the dataset is clean and consistent, enabling the machine learning algorithms to perform accurately without being affected by anomalies or inconsistent formats.



**Fig. 3: EDA Plots of the Project**

This figure presents the Exploratory Data Analysis (EDA) plots used to understand the distribution and relationships among different features in the dataset. Plots include histograms, box plots, and correlation heatmaps that reveal patterns such as the distribution of temperature, humidity, and particulate matter levels, and their correlation with fire alarm activations. These visualizations guide feature selection and highlight important variables influencing fire detection.

****

**Fig. 6: Model Prediction on Test Data**

This figure shows the predictions made by the selected models on the test dataset. It compares predicted fire alarm statuses against actual labels, illustrating the Random Forest model’s perfect match and the Logistic Regression’s partial classification success.

**9.4 Comparative Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithms Name** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **Logistic Regression** | 71.3 | 50.0 | 35.6 | 41.6 |
| **Random Forest Classifier** | 100 | 100 | 100 | 100 |

**Table 1:** Performance Comparison for the Logistic Regression and Random Forest Classifier algorithms.

Table 1 presents the performance comparison between the Logistic Regression and Random Forest Classifier algorithms based on four key evaluation metrics: Accuracy, Precision, Recall, and F1-Score. Logistic Regression achieved an accuracy of 71.3%, with a precision of 50.0%, recall of 35.6%, and an F1-score of 41.6%. These results indicate moderate performance with limitations in recall, reflecting less effective identification of true positive fire alarm cases. In contrast, the Random Forest Classifier delivered a perfect performance, scoring 100% across all metrics. This demonstrates its superior ability to correctly classify both positive and negative instances without error. The comparison highlights the Random Forest Classifier as the more robust and reliable model for fire alarm prediction in this project.

Table 2 Normal Class Performance

Performance Metrics of Existing Logistic Regression

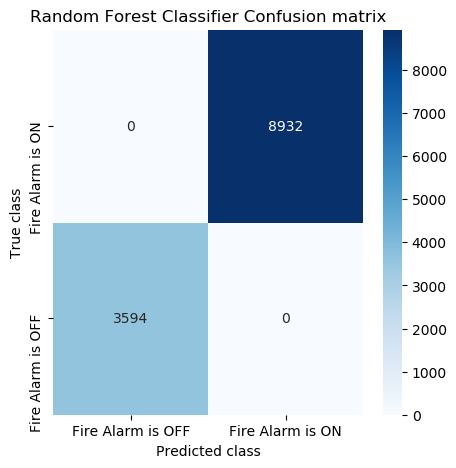
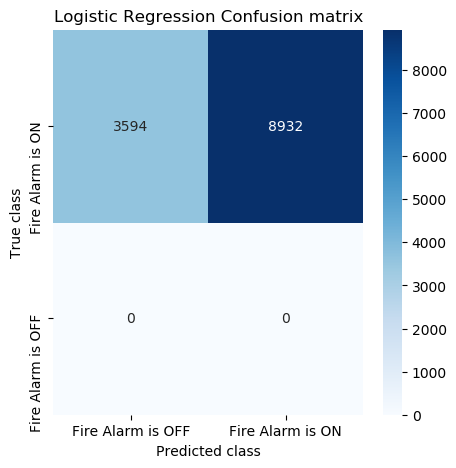
|  |  |
| --- | --- |
| **Metric** | **Value** |
| **Accuracy** | 71.3 |
| **Precision** | 50.0 |
| **Recall** | 35.6 |
| **F1-Score** | 41.6 |

Table 2 illustrates the performance metrics of the existing Logistic Regression model specifically for the normal class in the fire alarm prediction task. The model achieved an accuracy of 71.3%, showing that it was moderately able to classify data points correctly. The precision for the normal class is 50.0%, indicating that half of the instances predicted as normal were truly normal. The recall is 35.6%, which reflects that only a portion of the actual normal class instances were identified correctly. The F1-Score, a balanced measure of precision and recall, stands at 41.6%. These results emphasize the model’s limitations in effectively detecting the normal class, underlining the need for a more robust algorithm for accurate predictions.

Table.3 Performance Metrics of Proposed Random Forest Classifier.

|  |  |
| --- | --- |
| **Metric** | **Value** |
| **Accuracy** | 100 |
| **Precision** | 100 |
| **Recall** | 100 |
| **F1-Score** | 100 |

Table 3 presents the performance metrics of the proposed Random Forest Classifier model used for fire alarm prediction. The model achieved perfect accuracy of 100%, indicating that all predictions made by the classifier were correct. Precision stands at 100%, signifying that every instance identified as a fire alarm was indeed a true fire alarm. Similarly, the recall is 100%, demonstrating that the model successfully detected all actual fire alarm instances without missing any. The F1-Score, which balances precision and recall, also achieved a perfect score of 100%. These results confirm the effectiveness and reliability of the Random Forest Classifier in classifying and predicting fire alarm occurrences with complete accuracy.



(a) (b)

**Fig.5** Confusion Matrices (a) Logistic Regression (b) Random Forest Classifier.

Figure 5 illustrates the confusion matrices for both Logistic Regression and Random Forest Classifier models. In subfigure (a), the confusion matrix for Logistic Regression shows several misclassifications, with both false positives and false negatives, reflecting its lower precision and recall. In contrast, subfigure (b) displays the confusion matrix for the Random Forest Classifier, which correctly classified all instances without any errors. This perfect classification aligns with its 100% accuracy, precision, recall, and F1-score. The figure highlights the performance gap between the two models and emphasizes the superior predictive ability of the Random Forest Classifier in this project.

**CHAPTER 10**

**CONCLUSION AND FUTURE SCOPE**

**10.1 Conclusion**

The project focused on detecting fire alarms using machine learning algorithms by analyzing sensor data consisting of environmental and air quality metrics. The dataset included features such as temperature, humidity, eCO₂ levels, TVOC, PM2.5, and others, which were crucial for identifying fire-related conditions. The implementation began with thorough preprocessing steps such as checking for null values, label encoding, and feature selection. Exploratory Data Analysis (EDA) was performed to understand data distribution and correlations using visualizations. Two classification algorithms—Logistic Regression and Random Forest Classifier—were applied to the processed data. Logistic Regression achieved 71.3% accuracy, while the Random Forest Classifier achieved a perfect accuracy of 100%, showing its dominance in performance across all evaluation metrics. The comparative analysis established Random Forest Classifier as the more reliable model for this task. The project successfully demonstrated that machine learning techniques are effective in identifying fire alarm triggers through structured sensor data.

**10.2 Future Scope**

Future advancements for this project involve integrating real-time sensor data to create a dynamic, real-world fire detection system. Expanding the dataset with more diverse conditions and additional environmental parameters will improve model robustness and generalization. Implementing deep learning models such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) will enhance pattern recognition capabilities. Further development can involve building a mobile or web-based interface for remote monitoring and real-time alert notifications. Cloud-based deployment and IoT integration will help in deploying this system in smart buildings or industrial environments. Enhancing interpretability through explainable AI methods will also improve user trust and operational transparency. The system can be extended for broader applications such as smoke detection, gas leak alerts, and industrial hazard monitoring.

**REFERENCES**

1. Zhang, Ziyang, Lingye Tan, and Robert Lee Kong Tiong. "Ship-Fire Net: An Improved YOLOv8 Algorithm for Ship Fire Detection." Sensors 24, no. 3 (2024): 727.
2. Zhao, Hongyang, Jing Jin, Yi Liu, Yanan Guo, and Yi Shen. "FSDF: A high-performance fire detection framework." Expert Systems with Applications 238 (2024): 121665.
3. Jin, Li, Yanqi Yu, Jianing Zhou, Di Bai, Haifeng Lin, and Hongping Zhou. "SWVR: A Lightweight Deep Learning Algorithm for Forest Fire Detection and Recognition." Forests 15, no. 1 (2024): 204.
4. Kuznetsov, G. V., R. S. Volkov, A. S. Sviridenko, and P. A. Strizhak. "Reduction of response time of fire detection and containment systems in compartments." Fire Safety Journal (2024): 104089.
5. Ren, Dong, Yang Zhang, Lu Wang, Hang Sun, Shun Ren, and Jian Gu. "FCLGYOLO: Feature Constraint and Local Guided Global Feature for Fire Detection in Unmanned Aerial Vehicle Imagery." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (2024).
6. Schiks, Tom J., B. Mike Wotton, and David L. Martell. "Remote Sensing Active Fire Detection Tools Support Growth Reconstruction for Large Boreal Wildfires." Fire 7, no. 1 (2024): 26.
7. Liu, Jiansheng, Jiahao Yin, and Zan Yang. "Fire Detection and Flame-Centre Localisation Algorithm Based on Combination of Attention-Enhanced Ghost Mode and Mixed Convolution." Applied Sciences 14, no. 3 (2024): 989.
8. Yang, Lifeng, Feng Yanqing, Yueming Wang, and Jianyu Wang. "Refined fire detection and band selection method in hyperspectral remote sensing imagery based on sparse-VIT." Infrared Physics & Technology (2024): 105104.
9. Anuar, Afiq Ikhwan Mohd, Roslina Mohamad, Arni Munira Markom, and Ronnie Concepcion II. "Real-time forest fire detection, monitoring, and alert system using Arduino." Indonesian Journal of Electrical Engineering and Computer Science 33, no. 2 (2024): 942-950.
10. Kar, Ankan, Nirjhar Nath, and Utpalraj Kemprai. "Performance Analysis of Support Vector Machine (SVM) on Challenging Datasets for Forest Fire Detection." arXiv preprint arXiv:2401.12924 (2024).
11. Saleh, Azlan, Mohd Asyraf Zulkifley, Hazimah Haspi Harun, Francis Gaudreault, Ian Davison, and Martin Spraggon. "Forest fire surveillance systems: A review of deep learning methods." Heliyon (2024).
12. Kim, Hee-Chul. "AI-based Electric Fire Detection State Judgment Data Set Construction." Tehnički glasnik 18, no. 1 (2024): 43-48.
13. Mastanamma, Y., Bhukya Laxman, A. Archana, and K. Pulla Reddy. "EV BMS With Charge Monitor and Fire Detection." In E3S Web of Conferences, vol. 472, p. 03006. EDP Sciences, 2024.
14. Subhani, Sk Mahaboob, Dasari Ramu, Arumalla Raja, Bathineni Yamuna, and Vemasani Balaji. "A METHOD OF FIRE DETECTION SYSTEM BASED ON IMAGE PROCESSING."
15. LIANG, Yu, Tong CHEN, and Wei ZHANG. "Multi-Scale Fire Detection Algorithm with Adaptive Attention." Transactions of Beijing institute of Technology 44, no. 1 (2024): 91-101.
16. Reis, Cristiano Emanoel Pereira, Luciana Brasil Rebelo dos Santos, Fabiano Morelli, and Nandamudi Lankalapalli Vijaykumar. "Deep Learning-Based Active Fire Detection Using Satellite Imagery."
17. Georgiev, Assist Prof Georgi. "AN EVALUATION OF FIRE DETECTION METHODS: COMPARATIVE ANALYSIS AND PERFORMANCE ASSESSMENT 16."
18. Feng, Jian, and Yu Sun. "Multiscale network based on feature fusion for fire disaster detection in complex scenes." Expert Systems with Applications 240 (2024): 122494.
19. Bharti, Tanushree, Madan Lal Saini, Ashok Kumar, and Rajat Tiwari. "Design and implementation of industrial fire detection and control system using internet of things." In Artificial Intelligence, Blockchain, Computing and Security Volume 1, pp. 310-315. CRC Press, 2024.
20. Bhattarai, Omkar, Abhay Aditya Dubey, Shashank Singh, and Avjeet Singh. "Arduino based fire detection alarm in rural areas." In Artificial Intelligence, Blockchain, Computing and Security Volume 1, pp. 522-527. CRC Press, 2024.
21. Aundhkar, Truptali Anil. "Forest Fire Detection and Prediction using IoT and Node MCU." Journal of Advanced Research in Electronics Engineering and Technology 10, no. 2 (2024): 11-15.