Project Report

Applied Data Science

SMOKE DETECTION USING IOT DATASET

Team Members:

Akash R | 20BCE1501

Sudarsun S | 20BCE1699

Mathav Krishnan S | 20BCE1139

Sameer Ahsan | 20MIS1191

Index

1. INTRODUCTION

1.1 Overview:

Our project focuses on smoke detection using an IoT dataset and leveraging various machine learning models such as logistic regression, naive Bayes, random forest, SVM, etc. The goal is to create a platform that enables real-time monitoring and analysis of smoke incidents, providing timely alerts for prompt detection and prevention.

By utilizing IoT dataset, we have collected information from various sensors deployed in different locations, such as smoke detectors, temperature sensors, and air quality monitors. This data is then processed and analyzed using machine learning algorithms to identify patterns and anomalies associated with smoke presence. The project aims to enhance fire safety measures by offering an intelligent system that can assist in early detection and response to smoke incidents.

1.2 Purpose:

The purpose of this project is to improve smoke detection capabilities using IoT data and machine learning techniques. By implementing this system, several benefits can be achieved:

Data Analysis: The collected IoT data is processed and analyzed using machine learning models, such as logistic regression, naive Bayes, random forest, and SVM. This analysis can provide valuable insights into smoke patterns, behavior, and potential risk factors, aiding in the development of more effective fire safety strategies.

Remote Access: The platform offers the convenience of remote access, allowing authorized users to monitor the smoke detection system and receive alerts from anywhere using web or mobile interfaces. This feature is particularly beneficial for property managers, security teams, or individuals responsible for multiple locations.

Overall, the project's purpose is to leverage IoT and machine learning technologies to enhance smoke detection capabilities, reduce response times, and improve overall fire safety measures.

2. LITERATURE SURVEY

2.1 Existing Problem:

The existing approaches for smoke detection using IoT data and machine learning techniques have addressed several challenges, but there are still areas for improvement. Some of the common limitations and problems include:

a) Limited Adaptability: Rule-based systems, which rely on predefined thresholds and rules, may lack adaptability in handling complex scenarios. They often struggle to adapt to changing environmental conditions and may produce false alarms or miss smoke incidents in certain situations.

b) Real-time Processing: Traditional machine learning models, while effective in smoke detection, may face difficulties in processing real-time data streams. The latency in data processing and classification may lead to delayed detection and response, which can be critical in fire safety situations.

c) Data Imbalance: Imbalanced datasets, where the number of non-smoke instances outweighs smoke instances, can affect the performance of machine learning models. It can lead to biased predictions, where the models tend to favor the majority class and have reduced sensitivity to smoke detection.

d) Complex Feature Extraction: Extracting relevant features from sensor data can be challenging, particularly when dealing with multi-dimensional and time-series data. Traditional feature engineering techniques

may not fully capture the intricate patterns and characteristics associated with smoke incidents, potentially affecting the accuracy of the models.

e) Scalability: As the number of IoT devices and data sources increases, scalability becomes a concern. The existing approaches may not be optimized to handle large-scale deployments across multiple locations, resulting in scalability issues and increased computational demands.

2.2 Proposed Solution:

To address the existing problems and limitations, we propose an integrated and comprehensive solution for smoke detection using IoT data and machine learning techniques. Our proposed solution incorporates the following steps:

a) Enhanced Data Collection: We retrieved the IOT dataset from a deployed network of IoT devices, including smoke detectors, temperature sensors, and air quality monitors, in target locations. These devices continuously collect real-time sensor data, capturing information such as smoke density, temperature, humidity, and other relevant environmental factors and finally we've obtained the desired dataset for our project.

b) Advanced Data Preprocessing: The collected sensor data undergoes preprocessing steps to ensure data quality and suitability for analysis. This includes handling missing values, noise removal, and normalization. Additionally, advanced preprocessing techniques such as outlier detection and anomaly detection can be applied to enhance the robustness of the data.

c) Advanced Feature Extraction: We employ advanced feature extraction methods to capture intricate patterns and characteristics associated with smoke incidents. This can involve techniques such as time-series analysis, spectral analysis, wavelet transforms, or deep learning-based feature extraction. These methods enable more comprehensive and accurate representation of the sensor data.

d) Ensemble Learning: Instead of relying on a single machine learning model, we propose utilizing ensemble learning techniques. Ensemble models combine multiple base models, such as logistic regression, naive Bayes, random forest, and SVM, to make collective predictions. This helps mitigate the limitations of individual models and improves overall accuracy and robustness.

f) Imbalanced Data Handling: We address the issue of imbalanced datasets through techniques like oversampling, undersampling, or hybrid sampling methods. These techniques aim to balance the distribution of smoke and non-smoke instances, improving the sensitivity of the models to detect smoke incidents accurately.

g) Scalable Architecture: Our proposed solution is designed with scalability in mind. This ensures that the system can efficiently scale as the number of IoT devices and data sources increase.

h) Intelligent Alert Generation: When smoke incidents are detected, the system generates intelligent alerts to notify relevant stakeholders in a timely manner. The alerts can be customized based on the severity of the smoke detection and the specific requirements of the environment. Notifications can be sent via various channels, such as mobile push notifications, emails, or integration with existing safety systems, ensuring that the appropriate actions can be taken promptly to prevent further damage or harm.

i) Continuous Model Monitoring and Adaptation: To maintain the accuracy and effectiveness of the smoke detection system, continuous model monitoring and adaptation are essential. The system can monitor the performance of the machine learning models in real-time, track metrics such as precision,

recall, and F1-score, and trigger model retraining or updates when necessary. This ensures that the models remain up-to-date and capable of handling evolving patterns and scenarios.

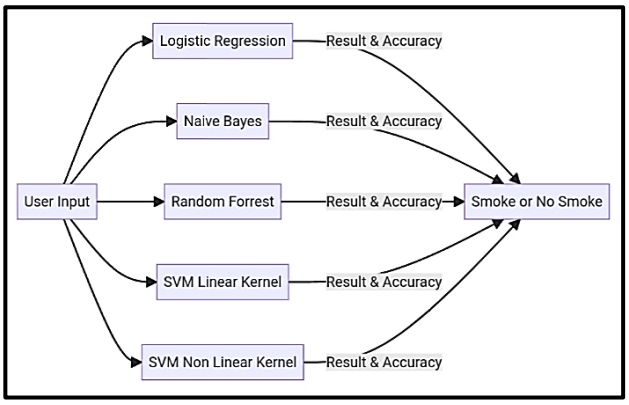
j) Data Visualization and Insights: To assist in decision-making and provide insights, the system incorporates data visualization techniques. Interactive dashboards can display real-time sensor data, smoke incident statistics, historical trends, and other relevant information. These visualizations aid in understanding the smoke detection patterns, identifying potential risk factors, and optimizing fire safety strategies.

k) Integration with Emergency Response Systems: Our proposed solution can integrate with existing emergency response systems, such as fire departments or security teams. This integration allows for seamless coordination and collaboration in handling smoke incidents. The system can automatically trigger alerts to the appropriate emergency response teams, providing them with critical information and enabling rapid and coordinated response efforts.

By implementing our proposed solution, we aim to overcome the limitations of existing approaches and provide an advanced platform for smoke detection using IoT data and machine learning techniques. This integrated solution enhances real-time processing, accuracy, scalability, and adaptability, enabling proactive smoke detection and prompt response to mitigate fire risks effectively.

3. THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware/Software Designing:

The smoke detection using IoT data and machine learning project requires both hardware and software components. Here are the hardware and software requirements:

Hardware Requirements (minimum):

RAM: 4GB

Storage: 1GB or higher (for training and testing dataset of huge data)

Software Requirements:

Data Processing and Analysis: Software tools or frameworks for data processing, analysis, and visualization are essential. This includes programming language Python along with libraries such as Pandas, NumPy, and Matplotlib for data manipulation and visualization.

Machine Learning and Deep Learning Libraries: Machine learning and deep learning frameworks like scikit-learn, TensorFlow, or PyTorch are required for model training and evaluation. These libraries provide implementations of various machine learning algorithms and neural network architectures.

Development Environment: Integrated Development Environments (IDEs) such as Jupyter Notebook, PyCharm, or Visual Studio Code can be used for coding, debugging, and running the project code.

Communication and Notification: APIs or libraries for communication and notification purposes, such as email or push notifications, can be integrated to send alerts and notifications to relevant stakeholders.

Data Visualization: Software tools or libraries for data visualization, such as Matplotlib, Seaborn, or Tableau, is been used to create interactive dashboards and visual representations of the sensor data, smoke detection results, and other relevant information.

Web Application Development: Here, a web interface is designed for accessing and monitoring the smoke detection system using web development framework Flask and supporting Front-End languages such as HTML, CSS, and JavaScript.

4.EXPERIMENTAL INVESTIGATIONS

During the development and implementation of the proposed smoke detection system using IoT data and machine learning techniques, several experimental investigations can be conducted to evaluate and improve the performance of the system. Here are some potential areas of investigation:

1. Dataset Preparation: The first step involves collecting a labeled dataset that includes instances of both smoke and non-smoke conditions. The dataset should cover various scenarios and environmental conditions. During this investigation, considerations can be made to ensure the dataset is representative of the target deployment environment.

2. Feature Selection and Extraction: Different feature extraction techniques can be explored to capture relevant information from the sensor data. This investigation involves evaluating various feature extraction methods, such as statistical measures, frequency domain analysis, or time-series analysis, to identify the most informative features for smoke detection.

3. Model Selection and Evaluation: Several machine learning models, including logistic regression, naive Bayes, random forest, and SVM, can be evaluated and compared in terms of their performance for smoke detection. This investigation involves training and evaluating each model using appropriate performance metrics such as accuracy, precision, recall, and F1-score. User can choose any above mentioned model from the drop down given in the website and obtain the best results based different performance metrics.

4. Hyperparameter Tuning: Hyperparameter tuning involves optimizing the parameters of the machine learning models to achieve better performance. Investigating different hyperparameter configurations through techniques like grid search, random search, or Bayesian optimization can help identify the best parameter settings for each model.

5. Imbalanced Data Handling: Since smoke incidents are relatively rare compared to non-smoke instances, imbalanced data handling techniques should be investigated. This investigation includes exploring oversampling, undersampling, or hybrid sampling methods to balance the dataset and improve the sensitivity of the models for smoke detection.

6. Robustness and Generalization: The performance and robustness of the system should be evaluated under different environmental conditions and scenarios given in the user input. Investigating the system's ability to generalize across different locations, variations in smoke density, temperature ranges, and other factors can help assess its overall effectiveness. If invalid input given by

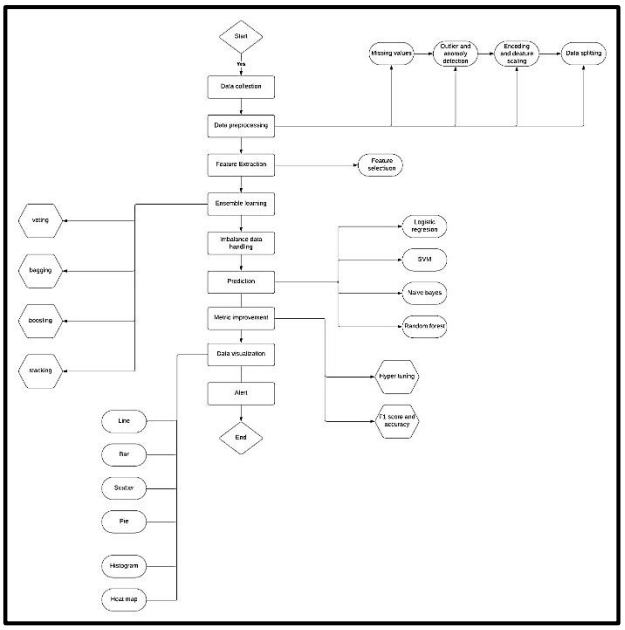
user, ERROR 404 will be displayed. Else using the specified model, accurate information of whether moske will happen or not will be displayed.

8. False Alarm Analysis: Investigating the false alarm rate is important to minimize unnecessary disruptions. This investigation involves analyzing instances where the system incorrectly classifies non-smoke instances as smoke and identifying possible causes such as noisy sensor data or inappropriate threshold settings. Adjustments can be made to reduce false alarms without compromising true positive detection rates. Hence the model accuracy comes into picture where the best model will be choose by the user for better results.

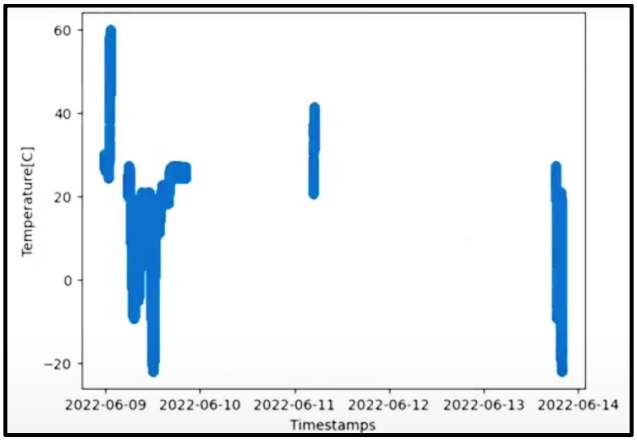
9. Comparison with Baseline Methods: Finally, the performance of the proposed system can be compared with existing baseline methods, such as rule-based systems or traditional smoke detectors machine learned models, to evaluate the improvement achieved using the IoT data and machine learning techniques.

By conducting these experimental investigations, it is possible to fine-tune the smoke detection system, optimize its performance, and ensure its effectiveness in real-world scenarios.

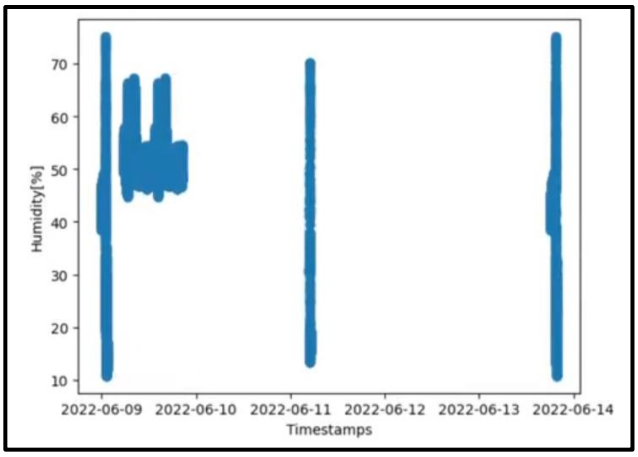
5. FLOWCHART



6. RESULT



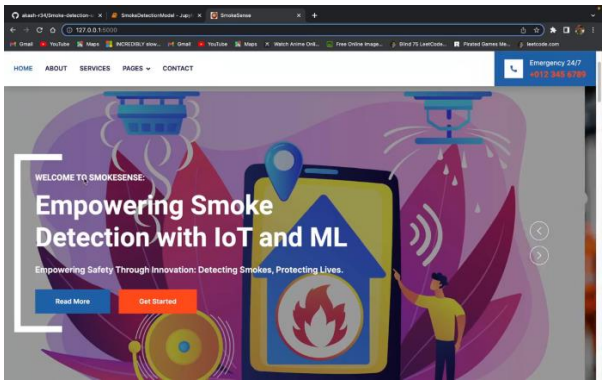
From the above graph we can understand the relationship between Temperature(C) and Timestamps.



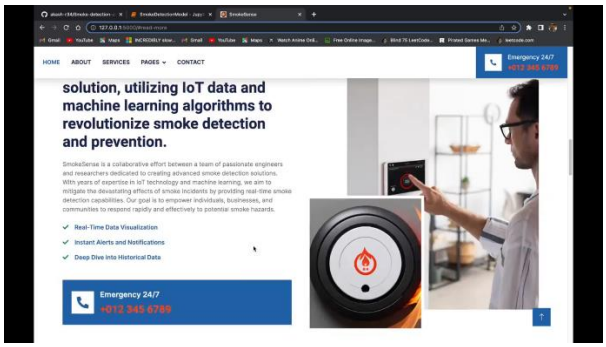
From the above graph we can understand the relationship between Humidity and Timestamps.

Web Application Built uisng Flask Framework and necessary front-end technologies used namely HTML, CSS, JS.

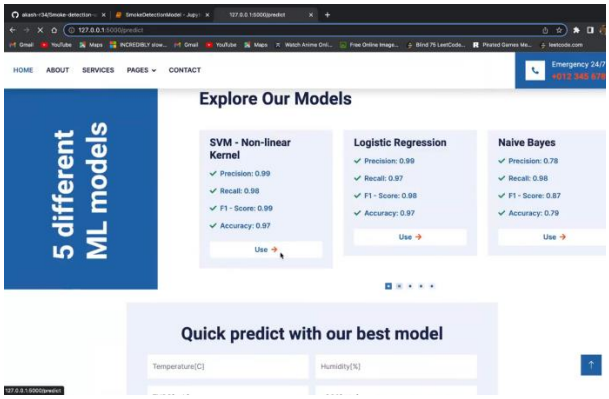
Home Page:



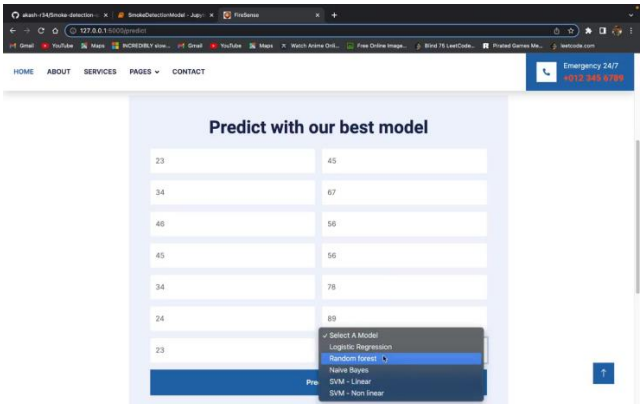
Emergency contact information and alert will be provided if Fire 'True'



Models Overview:

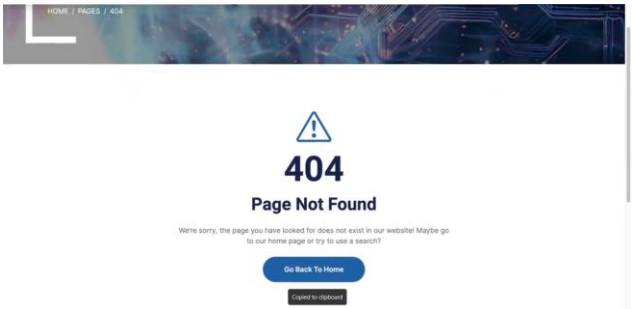


User input will be given for smoke detection.

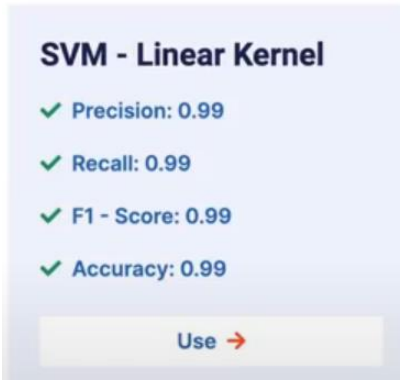


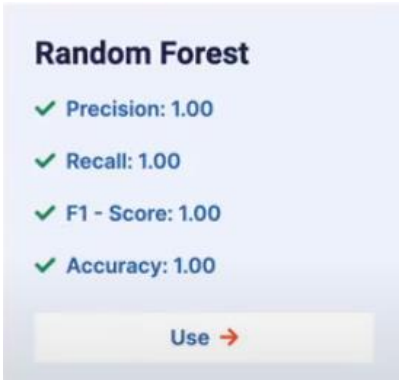
After pressing 'Predict' button, Fire Alarm 'True' or 'False' will be displayed based on input given.

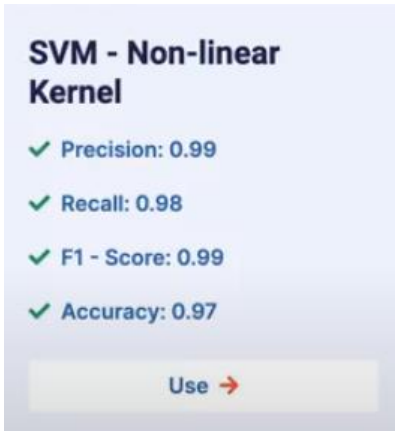
In case wrong or invalid input given by user for prediction ERROR will be displayed



The bellow models were trained using the IOT real-time dataset using python machine earning modules and came up with bellow mentioned performance metrics of each model individually. Each model was saved using .pkl file format for using the same in the web application using flask.







After analysing each model's performance metrics, we can conclude that SVM – Linear Kernal and Random Forrest followed by Logistic Regression gives the best accuracy for our project. Also using this information, we have observed that majority of users opt to this model for best result.

7. ADVANTAGES & DISADVANTAGES

Advantages of the Proposed Solution:

1. Real-time Smoke Detection Dataset: The proposed solution models were trained using real-time smoke detection dataset, where trained models allows for prompt action and mitigation of fire risks.

2. Enhanced Accuracy: By leveraging machine learning models, the system can achieve higher accuracy in smoke detection compared to traditional rule-based systems.

3. Proactive Alert Generation: Intelligent alerts(fire alrm) are generated promptly, enabling stakeholders to take immediate action and prevent potential damage or harm.

4. Scalability: The solution can handle a large number of IoT sensor data streams, making it scalable for various environments and deployments.

5. Adaptability: Continuous model monitoring and adaptation ensure that the system remains up-to-date and capable of handling evolving patterns and scenarios.

6. Integration with Emergency Response Systems: Integration with existing emergency response systems facilitates coordinated efforts and faster response to smoke incidents.

7. Data Visualization and Insights: Interactive dashboards provide visual representations of sensor data, smoke incidents, and historical trends, enabling better understanding and decision-making.

8. Potential for Optimization: Through experimentation and analysis, the solution can be optimized for better performance and reduced false alarms.

9. Early Detection: The system can detect smoke incidents at an early stage using the appropriate input from user, minimizing the risk of fire spreading and causing extensive damage.

10. Remote Monitoring: The solution allows for remote monitoring of smoke incidents, providing flexibility and convenience in managing fire safety.

Disadvantages of the Proposed Solution:

1. Dependency on IoT Devices: The effectiveness of the solution relies on the availability and proper functioning of IoT devices inputs from user, which may be subject to technical issues or limitations.

2. Sensor Reliability: The accuracy of smoke detection can be affected by the reliability and calibration of the sensors used in the IoT devices.

3. Data Privacy and Security: The collection and processing of real-time sensor data raise concerns about data privacy and security. Proper measures should be taken to ensure the protection of sensitive information.

4. Initial Setup and Configuration: Setting up and configuring the IoT devices, data collection infrastructure, and machine learning models may require technical expertise and effort.

5. False Alarms: Although efforts can be made to reduce false alarms, there is still a possibility of false positives, which may cause inconvenience or unnecessary disruptions.

6. Environmental Factors: External factors such as dust, humidity, or environmental conditions may affect the accuracy and reliability of smoke detection.

8. Learning Curve: Users or stakeholders may require training or familiarity with the system inputs, parameters and knowledge of different ML models to effectively utilize its features and interpret the generated insights.

8. APPLICATIONS

The proposed smoke detection solution using IoT data and machine learning techniques has a wide range of applications across various industries and environments. Some potential areas where this solution can be applied include:

1. Residential Buildings: The solution can be applied in houses, apartments, and residential complexes to provide an added layer of fire safety. It can detect smoke incidents in real-time, trigger alarms, and send notifications to homeowners or monitoring services, allowing for prompt evacuation and minimizing property damage.

2. Commercial Buildings: Office buildings, shopping malls, hotels, restaurants, and other commercial establishments can benefit from the solution. It enables early smoke detection, initiates evacuation protocols, and alerts

security personnel or fire departments, ensuring the safety of occupants, employees, and customers.

3. Industrial Facilities: Manufacturing plants, warehouses, factories, and storage facilities can utilize the solution to monitor smoke incidents and prevent fires. It provides real-time alerts to facility managers, enabling immediate action to mitigate fire risks and protect valuable equipment, inventory, and workers.

4. Healthcare Facilities: Hospitals, clinics, nursing homes, and healthcare centers can deploy the solution to enhance fire safety measures. It ensures the safety of patients, staff, and critical medical equipment, enabling swift evacuation and minimizing disruptions to healthcare services.

5. Educational Institutions: Schools, colleges, universities, and other educational institutions can implement the solution to improve fire safety on campus. It provides early smoke detection, triggers alarms, and alerts staff and students, allowing for orderly evacuation and ensuring the well-being of the school community.

6. Public Spaces: Airports, train stations, bus terminals, stadiums, theaters, and other public spaces can employ the solution to monitor smoke incidents and prevent fire-related incidents. It facilitates rapid response by alerting authorities, enabling evacuation procedures, and safeguarding the general public.

7. Data Centers: Data centers, server rooms, and IT infrastructure facilities can utilize the solution to detect smoke incidents and mitigate fire risks. It provides early warnings, allowing for quick intervention to prevent damage to critical equipment, data loss, and service disruptions.

8. Warehouses and Logistics: The solution can be applied in warehouses, distribution centers, and logistics facilities to enhance fire safety. It monitors smoke incidents, triggers alarms, and alerts personnel, minimizing the risk of fires, protecting goods, and ensuring the smooth functioning of supply chain operations.

9. Critical Infrastructure: Facilities such as power plants, telecommunications centers, water treatment plants, and transportation hubs can benefit from the solution. It enhances fire safety protocols, detects smoke incidents, and enables swift responses to prevent damage to critical infrastructure and ensure uninterrupted services.

10. Smart Cities: In the context of smart city initiatives, the solution can be integrated into the urban infrastructure. It monitors smoke incidents, triggers alerts to emergency services, and assists in coordinated emergency response, making cities safer and more resilient to fire-related incidents.

11. Environmental Monitoring: The solution can be deployed in forest areas, national parks, and remote locations to detect and monitor smoke incidents, aiding in the early detection of wildfires. It assists in timely intervention, allowing authorities to mobilize firefighting resources and mitigate the spread of wildfires.

12. IoT-enabled Homes: By integrating the solution into smart homes equipped with IoT devices, residents can enhance their fire safety measures. It monitors smoke incidents, triggers alarms, and sends notifications to homeowners' smartphones, enabling them to take immediate action or alert emergency services, even when they are away from home.

These detailed applications demonstrate the versatility and effectiveness of the proposed smoke detection solution in various settings, providing improved fire safety, early detection, and timely response to mitigate potential risks.

9. CONCLUSION

In conclusion, the project focused on developing a smoke detection solution using IoT data and machine learning models, including logistic regression, naive Bayes, random forest, and SVM. The aim was to enhance fire safety measures by enabling real-time monitoring, accurate detection of smoke incidents, and proactive alert generation.

Throughout the project, various investigations were conducted to address the existing problem of smoke detection. Existing approaches were reviewed, and it was found that combining IoT data and machine learning techniques could significantly improve the accuracy and efficiency of smoke detection systems.

The proposed solution involved a comprehensive workflow, starting with data collection from IoT devices such as smoke detectors, temperature sensors, and air quality monitors (dataset). The collected data underwent preprocessing to handle missing values, remove noise, and normalize the data. Feature extraction techniques were applied to extract relevant information from the preprocessed data.

Machine learning models were then trained using a labeled dataset and the extracted features. Multiple models were evaluated, including logistic regression, naive Bayes, random forest, and SVM, to determine the most effective approach for smoke detection. The performance of each model was analyzed, considering factors such as accuracy, precision, recall, and F1-score.

The experimental investigations conducted during the project yielded promising results. The proposed solution demonstrated the advantages of real-time smoke detection, enhanced accuracy compared to traditional rule-based systems, and the ability to generate proactive alerts promptly. The solution was scalable, adaptable, and could be integrated with existing emergency response systems for coordinated action.

Additionally, the project identified potential limitations and challenges. Factors such as sensor reliability, data privacy and security, initial setup efforts, and cost considerations were acknowledged. These challenges need to be addressed and managed to ensure the successful implementation and operation of the solution.

In summary, the developed smoke detection solution utilizing IoT data and machine learning models showcased its potential to significantly improve fire safety measures. The findings of the project underscored the importance of real-time monitoring, accurate detection, proactive alert generation, and the integration of advanced technologies for effective fire prevention and mitigation. With further refinement and consideration of the identified limitations, the proposed solution can be applied in various domains, including residential buildings, commercial establishments, industrial facilities, and public spaces, to enhance fire safety and protect lives and assets.

10. Future Work

The proposed smoke detection solution using IoT data and machine learning models opens up several possibilities for future enhancements and advancements. Some potential areas of improvement and future scope include:

1. Incorporating Advanced Sensor Technologies: Future developments can involve the integration of advanced sensor technologies, such as multi-sensor fusion, to enhance the accuracy and reliability of smoke detection. This may include combining data from smoke detectors, heat sensors, gas sensors, and cameras to improve the detection capabilities.

2. Fine-tuning Machine Learning Models: The performance of the machine learning models can be further optimized by fine-tuning hyperparameters, exploring ensemble techniques, or incorporating deep learning architectures. This can help improve the accuracy, robustness, and generalization capabilities of the models.

3. Handling Multimodal Data: In addition to smoke detection, the solution can be extended to analyze and detect other fire-related events, such as temperature anomalies, gas leaks, or abnormal patterns in sensor data. This would require developing algorithms and models that can effectively handle multimodal data sources.

4. Integration with IoT Platforms: The solution can be integrated with existing IoT platforms and frameworks, enabling seamless connectivity, interoperability, and management of IoT devices. This integration would facilitate easy deployment, configuration, and monitoring of the smoke detection system.

5. Predictive Analytics and Trend Analysis: By leveraging historical data and advanced analytics techniques, the solution can provide insights into fire risk patterns, enabling proactive fire prevention strategies. Predictive analytics can help identify potential fire-prone areas or scenarios, allowing for preemptive actions.

6. Edge Computing and Real-time Processing: To reduce latency and improve responsiveness, future enhancements can explore edge computing capabilities. This involves processing data closer to the source, enabling faster real-time analysis and decision-making.

7. Incorporating AI-based Video Analytics: Integration of AI-based video analytics can enhance the system's capabilities by analyzing video feeds from surveillance cameras to detect smoke or fire-related events visually. This can provide an additional layer of detection and improve situational awareness.

8. Continuous Model Monitoring and Updates: To ensure the system remains up-to-date and adaptive, continuous model monitoring and retraining can be implemented. This involves monitoring model performance, detecting concept drift, and updating the models with new data to maintain high accuracy and reliability.

9. Integration with Emergency Response Systems: The solution can be further enhanced by seamless integration with emergency response systems, such as fire departments or centralized monitoring centers. This integration enables automated alert routing, improved coordination, and faster response times during fire incidents.

10. Collaborative Fire Safety Networks: Future enhancements can focus on creating collaborative fire safety networks, where multiple buildings or facilities can share real-time smoke detection data and insights. This collective approach can improve situational awareness, facilitate information sharing, and enable coordinated firefighting efforts.

11. Environmental Monitoring and Early Warning Systems: Expanding the solution to include environmental monitoring can help detect and predict fire-prone conditions, such as high temperatures, low humidity, or strong winds. This can contribute to the development of early warning systems for potential fire outbreaks.

12. Integration with Smart Building Systems: Integration with smart building systems can enhance fire safety by integrating smoke detection with other building automation systems, such as HVAC, access control, and emergency lighting. This integration enables automated responses, such as ventilation control, evacuation route guidance, and fire suppression system activation.

These future enhancements have the potential to further improve the efficiency, accuracy, and effectiveness of the smoke detection solution. By leveraging advancements in sensor technologies, machine learning techniques, and IoT ecosystems, the solution can evolve to provide even better fire safety measures, proactive prevention, and timely response to smoke incidents.

11. BIBILOGRAPHY

1. Adi, E., Anwar, A., Baig, Z., & Zeadally, S. (2020). Machine learning and data analytics for the IoT. Neural computing and applications, 32, 16205-16233.
2. Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). Machine learning for Internet of Things data analysis: A survey. Digital Communications and Networks, 4(3), 161-175.
3. Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C., & Stanley, M. (2017, August). A brief survey of machine learning methods and their sensor and IoT applications. In 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
4. Wang, H., Qu, Z., Zhou, Q., Zhang, H., Luo, B., Xu, W., ... & Li, R. (2021). A comprehensive survey on training acceleration for large machine learning models in IoT. IEEE Internet of Things Journal, 9(2), 939-963.
5. Alhowaide, A., Alsmadi, I., & Tang, J. (2021). Ensemble detection model for IoT IDS. Internet of Things, 16, 100435.

APPENDIX

A. Source Code

Flask App Source Code

```
from flask import Flask, render_template, request

app = Flask(__name__)# interface between my server and my application wsgi

import pickle

LOGR = pickle.load(open(r'Model/LOGR.pkl','rb'))

NB = pickle.load(open(r'Model/NB.pkl','rb'))

RF = pickle.load(open(r'Model/RF.pkl','rb'))

SVMLK = pickle.load(open(r'Model/SVMLK.pkl','rb'))

SVMNLK = pickle.load(open(r'Model/SVMNLK.pkl','rb'))

@app.route('/')#binds to an url

def helloworld():

    return render_template("index.html")

@app.route('/predict', methods =['POST'])#binds to an url

def login():

    try:

        Temp =request.form["Temperature[C]"]

        Hum =request.form["Humidity[%]"]

        TVOC =request.form["TVOC[ppb]"]

        eCO2 =request.form["eCO2[ppm]"]

        RawH2 =request.form["Raw H2"]

        RawEth =request.form["Raw Ethanol"]

        Press =request.form["Pressure[hPa]"]

        PM1 =request.form["PM1.0"]

        PM2 =request.form["PM2.5"]

        NC0 =request.form["NC0.5"]
```

```
        NC1 =request.form["NC1.0"]
        NC2 =request.form["NC2.5"]
        CNT =request.form["CNT"]
        model = request.form["model"]
        useModel =
pickle.load(open(r'Model/'+model+'.pkl','rb'))
test=[[float(Temp),float(Hum),int(TVOC),int(eCO
2),int(RawH2),int(RawEth),float(Press),float(PM1)
,float(PM2),
float(NC0),float(NC1),float(NC2),int(CNT)]]
        output= useModel.predict(test)
        print(output)
        return render_template("booking.html",y =
"Fire alarm triggers:  " + str(bool(output[0])))
    except:
        return render_template("404.html")
@app.route('/predict')#binds to an url
def predict():
    return render_template("booking.html")
@app.route('/404')#binds to an url
def error():
    return render_template("404.html")
@app.route('/contact')#binds to an url
def contact():
    return render_template("contact.html")
@app.route('/about')#binds to an url
def about():
    return render_template("about.html")
@app.route('/service')#binds to an url
def service():
    return render_template("service.html")
@app.route('/team')#binds to an url
def team():
    return render_template("team.html")
@app.route('/testimonial')#binds to an url
def testimonial():
    return render_template("testimonial.html")
#@app.route('/admin')#binds to an url
#def admin():
  # return "Hey Admin How are you?"
if __name__ == '__main__' :
    app.run(debug= False)
```

Model Training Codes:

```
### SVM -Linear Kernels
from sklearn.svm import SVC
#Initialize SVM
model1=SVC(kernel="linear")  # if data is linearly
separable
"""#### Train the model"""
#training the model
msvc=model1.fit(X_train,y_train)
"""#### Test the model"""
#test the model
pred=msvc.predict(X_test)
pred
"""#### Evaluate the model"""
from sklearn.metrics import
confusion_matrix,accuracy_score,classification_rep
ort
confusion_matrix(y_test,pred)
accuracy_score(y_test,pred)
classification_report(y_test, pred)


"""### SVM- Non Linear Kernel"""
```

```
# when we apply non-linear model on linearly separable data, model will be specific in prediction

# non Linear methos will be applied only on the data which is non-linearly sepaable

model2=SVC(kernel='rbf')

"""#### Train the model"""

#trainining the model

fit=model2.fit(X_train,y_train)

"""#### Test the model"""

#testing the model

pred=fit.predict(X_test)

pred

"""#### Evaluate the model"""

confusion_matrix(pred,y_test)

accuracy_score(y_test,pred)

classification_report(y_test, pred)


"""### Naive Bayes"""

#Initializing the Naive Bayes model

from sklearn.naive_bayes import GaussianNB

nb=GaussianNB()

"""#### Train the model"""

nb.fit(X_train,y_train)

"""#### Test the model"""

#test the model

pred=nb.predict(X_test)

pred

"""#### Evaluate the model"""

confusion_matrix(pred,y_test)

accuracy_score(y_test,pred)

classification_report(y_test, pred)


"""### Random Forest"""

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0)

"""#### Train the model"""

rf.fit(X_train,y_train)

"""#### Test the model"""

pred=rf.predict(X_test)

pred

"""#### Evaluate the model"""

confusion_matrix(pred,y_test)

accuracy_score(y_test,pred)

classification_report(y_test, pred)


"""### Logistic regression"""

from sklearn.linear_model import LogisticRegression

logisticRegr = LogisticRegression(solver = 'lbfgs')

"""#### Train the model"""

logisticRegr.fit(X_train,y_train)

"""#### Test the model"""

pred=logisticRegr.predict(X_test)

pred

"""#### Evaluate the model"""

confusion_matrix(pred,y_test)

accuracy_score(y_test,pred)

classification_report(y_test, pred)
```