

## **Project Report -TEAM 592381**

### **PROJECT TITLE-Detect smoke with the help of IOT Data**

### **and Trigger a fire Alarm Team Members:**

**KOMMAREDDY LEKHANA-21BCE1563**

**VIGRAHALA RAJESHRI-21BCE1921**

**CHENNAREDDYGARI RUSHITHA-21BCE5460**

#### **1. INTRODUCTION**

- 1.1 Project Overview
- 1.2 Purpose

#### **2. LITERATURE SURVEY**

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

#### **3. IDEATION & PROPOSED SOLUTION**

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming

#### **4. REQUIREMENT ANALYSIS**

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

#### **5. PROJECT DESIGN**

- 5.1 Data Flow Diagrams & User Stories
- 5.2 Solution Architecture

#### **6. PROJECT PLANNING & SCHEDULING**

- 6.1 Technical Architecture
- 6.2 Sprint Planning & Estimation
- 6.3 Sprint Delivery Schedule

#### **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema (if Applicable)

#### **8. PERFORMANCE TESTING**

- 8.1 Performace Metrics

## 9. **RESULTS**

### 9.1 Output Screenshots

## 10. **ADVANTAGES & DISADVANTAGES**

## 11. **CONCLUSION**

## 12. **FUTURE SCOPE**

## 13. **APPENDIX** Source Code GitHub

& Project Demo Link

## **PROJECT: Detect Smoke With The Help Of IOT Data And Trigger A Fire Alarm**

### **1.INTRODUCTION 1.1**

#### **PROJECT OVERVIEW:**

The Smoke Detection using IoT and Alarm Triggering project aims to detect smoke in real-time using IoT technology and trigger an alarm. The system will utilize highly sensitive smoke sensors and wireless connectivity to transmit data to a central hub or cloud platform for analysis. Upon detecting smoke, the system will activate an alarm to promptly alert individuals nearby. Users will have access to a mobile or web interface to remotely monitor the system status, receive real-time notifications, and control the alarm. This innovative solution offers early detection, timely alerts, remote accessibility, and cost-effectiveness, ultimately enhancing safety measures and reducing the risk of fire accidents. The use of IoT technology and wireless connectivity reduces the need for complex wiring installations, making the system easy to deploy. The alarm triggering mechanism ensures that individuals are immediately notified of a potential fire hazard, enabling them to take necessary precautions or evacuate if needed. The system provides an efficient solution for smoke detection and alarm triggering, ultimately enhancing safety measures. The project offers several benefits, including early detection, timely alerts, remote accessibility, and cost-effectiveness. The Smoke Detection using IoT and Alarm Triggering project provides an innovative solution for efficient smoke detection and alarm triggering. The project's objectives include developing and training machine learning models on IoT data, evaluating their performance, integrating the system with fire alarm mechanisms, and validating the effectiveness of the solution.

#### **1.2 PURPOSE**

The purpose of the Smoke Detection using IoT and Alarm Triggering project is to provide an innovative solution for efficient smoke detection and alarm triggering. By leveraging IoT technology, this system offers real-time monitoring, timely alerts, and remote accessibility, ultimately enhancing safety measures and reducing the risk of fire accidents. The use of highly sensitive smoke sensors and wireless connectivity ensures early detection of smoke, allowing for prompt action to prevent potential fire accidents. The alarm triggering mechanism ensures that individuals are immediately notified of a potential fire hazard, enabling them to take necessary precautions or evacuate if needed. The system provides an efficient solution for smoke detection and alarm triggering, ultimately enhancing safety measures. The project aims to offer several benefits, including early detection, timely alerts, remote accessibility, and costeffectiveness.

### **2.LITERATURE SURVEY:**

**2.1 EXISTING PROBLEM:** One of the existing problems with smoke detection using IoT and triggering an alarm in industries is the lack of reliable and efficient systems for early smoke detection. Traditional smoke detection systems are often limited in their capabilities and may not provide real-time monitoring or timely alerts. This can lead to delayed response times and increased risk of fire accidents. Additionally, some systems may require complex wiring

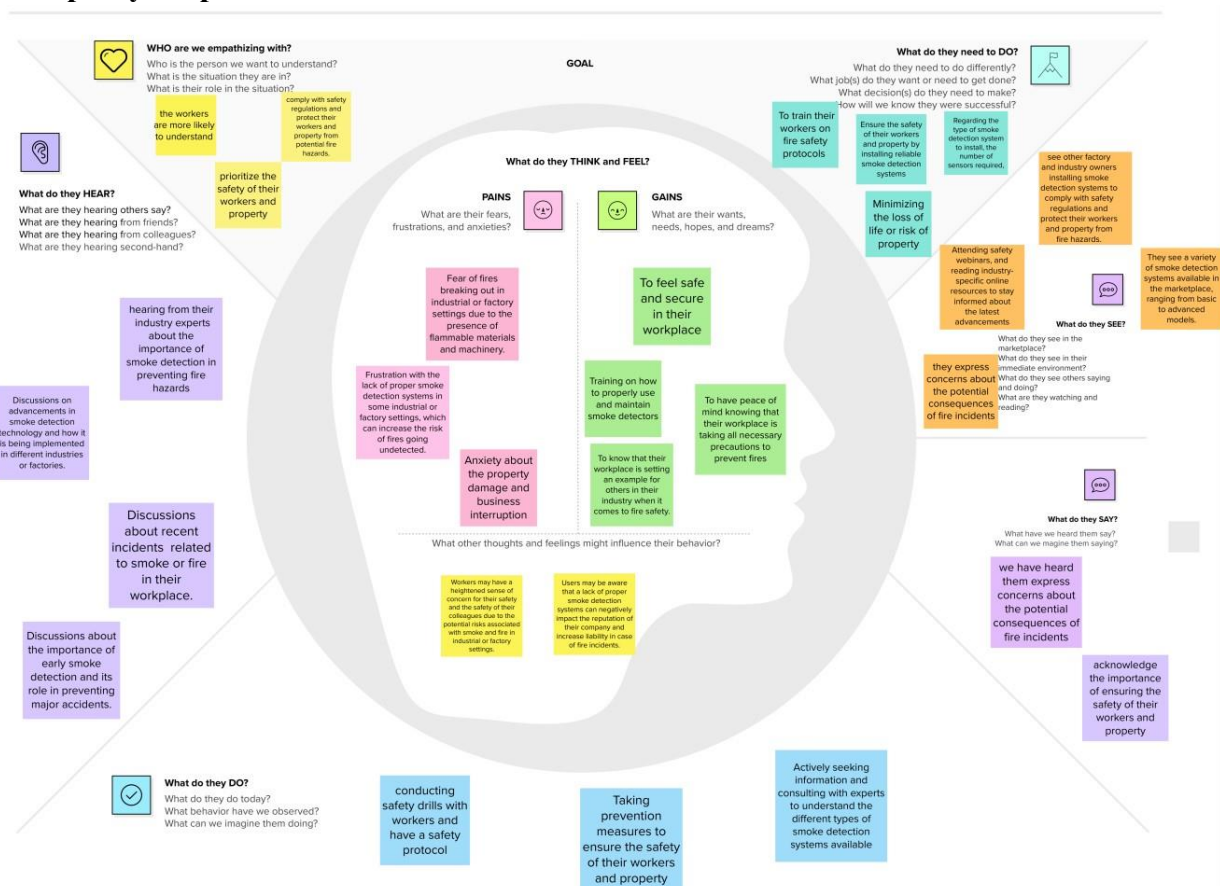
installations, making them costly and difficult to deploy. Furthermore, some systems may not offer remote accessibility, making it challenging to monitor the system status and receive realtime notifications. These limitations can pose significant challenges for industries that require reliable and efficient smoke detection systems to ensure the safety of their employees and assets.

## 2.2 REFERENCES:

**2.3 PROBLEM STATEMENT DEFINITION:** The problem statement involves creating an smoke detection system with IOT data system detect smoke and activate a fire alarm when smoke is present. It requires the use of sensors to monitor environmental conditions, such as smoke concentration, temperature, and humidity. The collected data is processed and analyzed in realtime to determine the presence of smoke. When smoke is detected, the system triggers a fire alarm, possibly through audible alerts or notifications to relevant parties. It should be designed with reliability, compliance with safety standards, and scalability in mind to suit various deployment scenarios.

## 3. IDEATION AND PROPOSED SOLUTION:

### 3.1 Empathy Map Canvas:



### 3.2 Ideation & Brainstorming:



## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare

🕒 1 hour to collaborate

👤 2-8 people recommended



### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes



#### A Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.



#### B Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.



#### C Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →



### Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

#### PROBLEM

Safety is a crucial consideration in the design of industrial buildings to safeguard against the loss of life, property damage, and environmental impact.



#### Key rules of brainstorming

To run an smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

#### TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

#### Person 1

Integrating the system with IOT

Mobile app to monitor and control the system

Using advanced optical sensors for smoke detection

#### Person 2

using temperature and humidity to detect smoke

early warning systems that can detect the early signs of a potential fire

real-time data sharing between smoke detection systems in different industrial facilities

#### Person 3

Using advanced optical sensors for smoke detection

using temperature and humidity to detect smoke

AI based sensors

3

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

#### TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas or themes within your mural.

Using temperature and humidity to detect smoke

Mobile app to monitor and control the system

Early warning systems that can detect the early signs of a potential fire

4

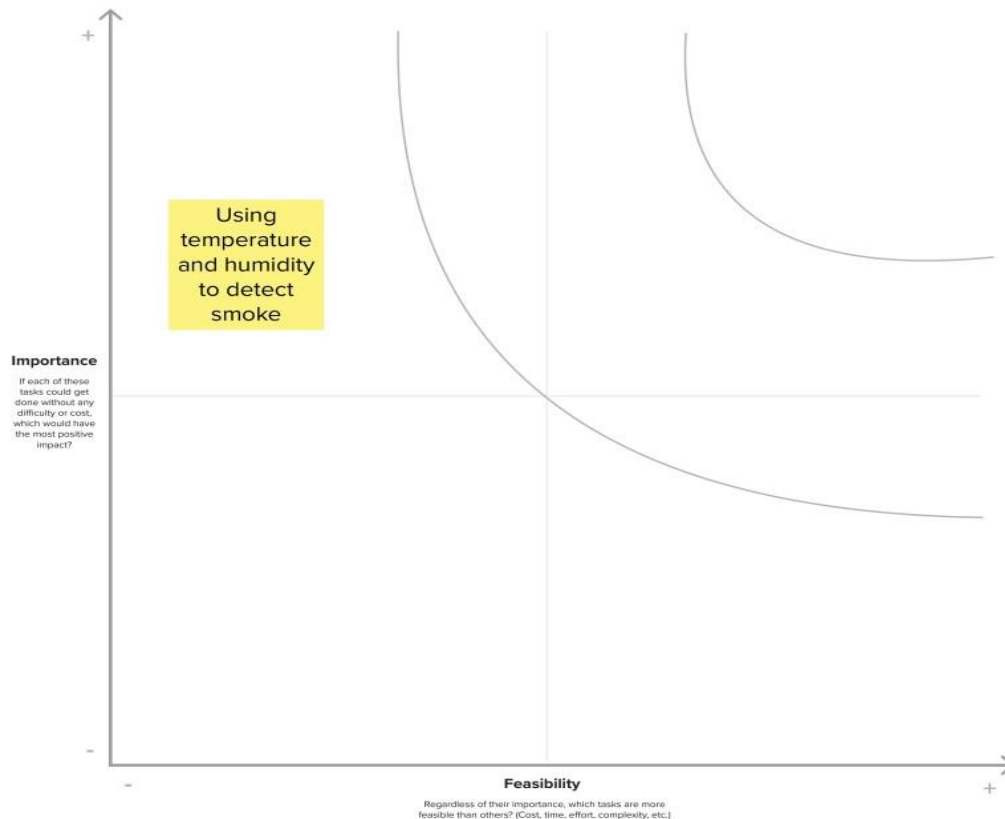
**Prioritize**

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

**TIP**

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H** key on the keyboard.



## 4. REQUIREMENT ANALYSIS:

### 4.1 FUNCTIONAL REQUIREMENTS:

**Real-time Smoke Detection:** The system should be able to detect the presence of smoke in real-time using highly sensitive smoke sensors.

**Data Analysis and Reporting:** The system should analyze the collected data to determine the presence and intensity of smoke and provide detailed reports for further analysis and decisionmaking.

**Compliance with Safety Standards:** The system should adhere to relevant safety standards and regulations to ensure its effectiveness and reliability in industrial environments.

**Alarm Triggering Mechanism:** Upon detecting smoke, the system should activate an alarm, such as a siren or notification, to promptly alert individuals in the industrial setting.

**Remote Monitoring and Control:** Users should have access to a mobile application or web interface that allows them to remotely monitor the system status, receive real-time notifications, and control the alarm.

#### 4.2 NON-FUNCTIONAL REQUIREMENTS:

**Reliability:** The system should be highly reliable, with minimal downtime or false alarms, to ensure the safety of employees and assets.

**Scalability:** The system should be scalable to accommodate the changing needs of the industry and the growing number of sensors and devices.

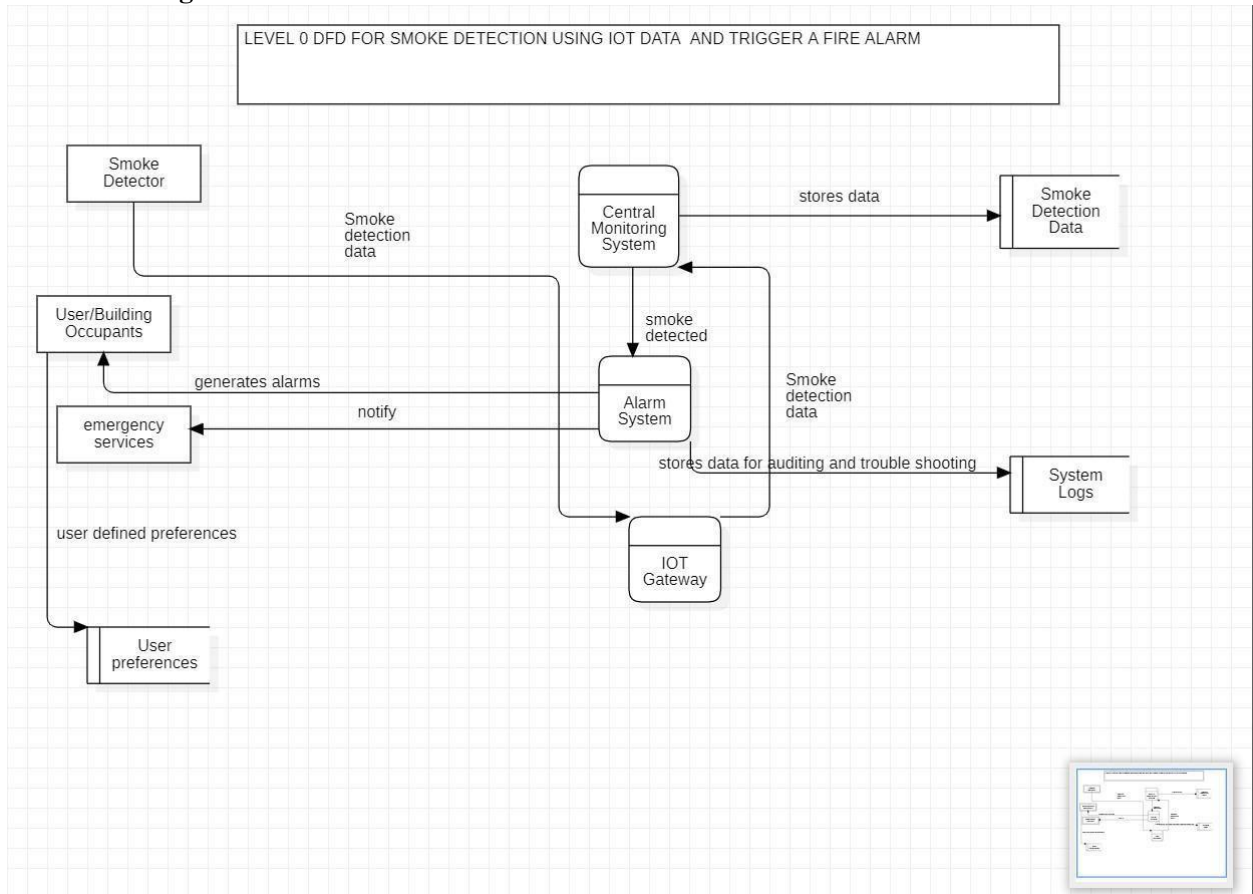
**Security:** The system should be secure, with robust authentication and encryption mechanisms, to prevent unauthorized access or tampering of data.

**Compatibility:** The system should be compatible with a wide range of devices, sensors, and platforms to enable seamless integration and interoperability.

**Usability:** The system should be user-friendly, with a simple and intuitive interface, to enable easy monitoring and control of the system.

### 5.PROJECT DESIGN

#### 5.1 Data Flow Diagrams & User Stories



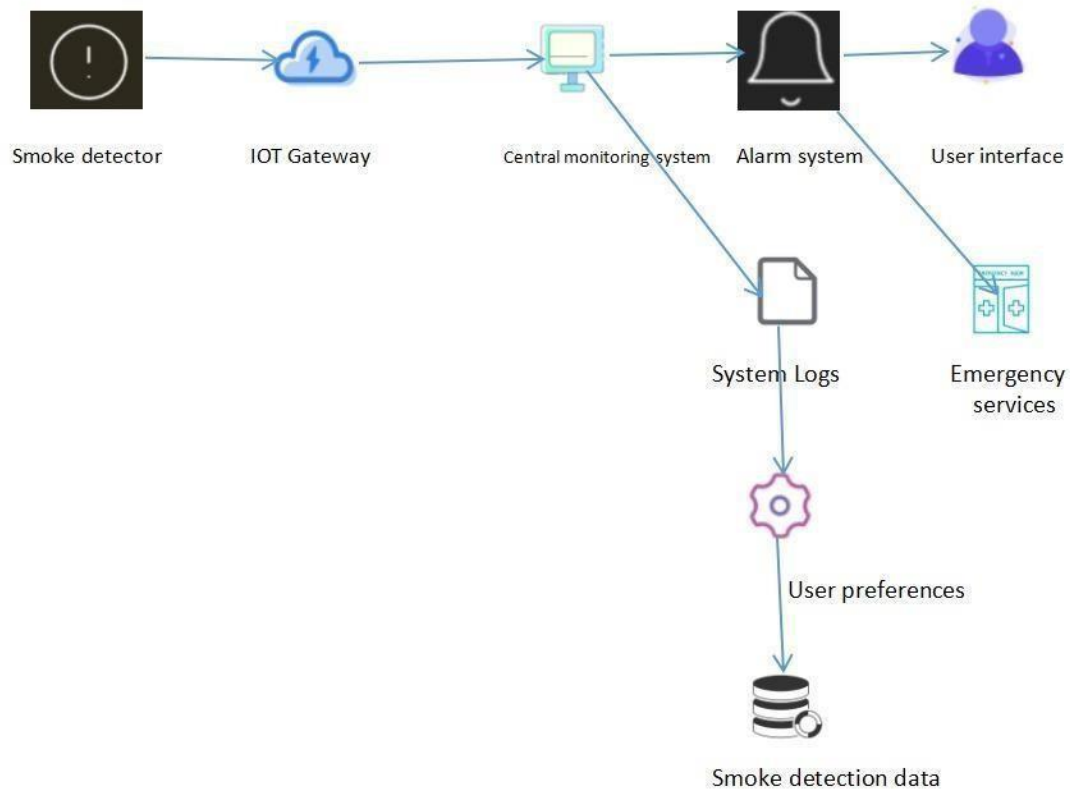


### User Stories

Use the below template to list all the user stories for the product.

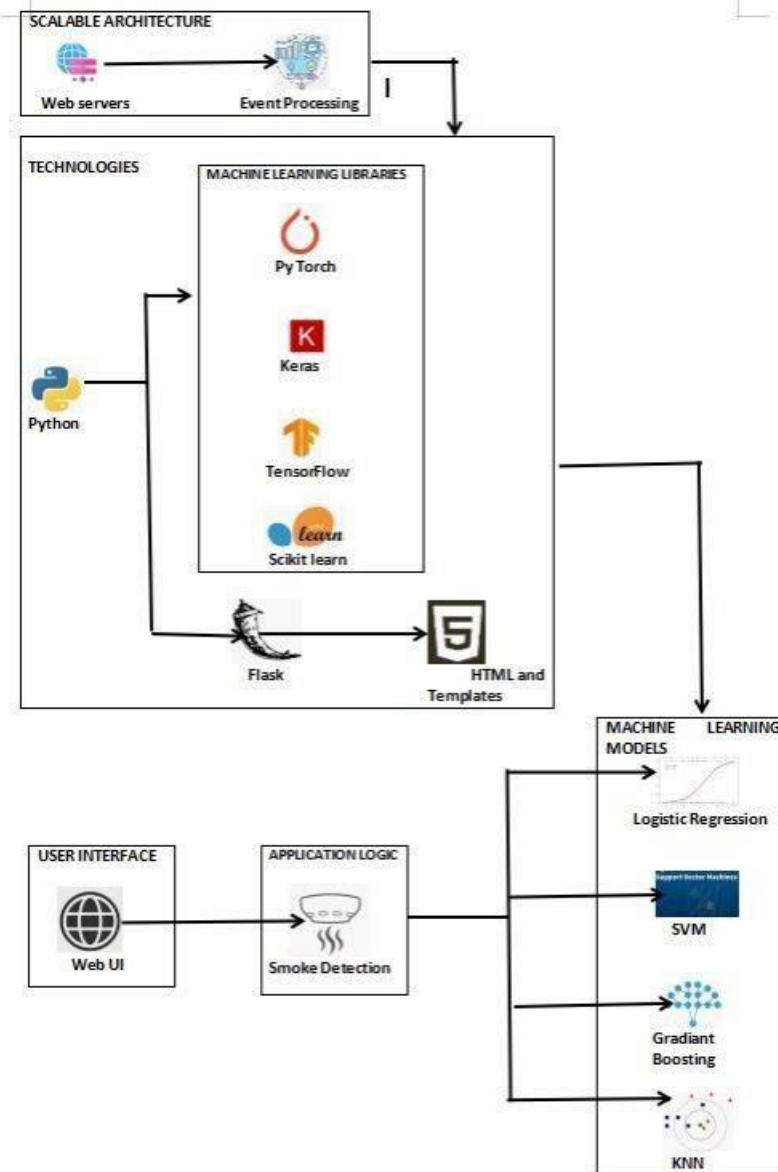
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Employee	Smoke Detection System	USN-1	As an employee, I want the system to provide real-time monitoring of the industrial facility's air quality, so I can work in a safe environment.	Smoke data received from IoT devices.	High	Sprint-1
		USN-2	As an employee, I want the system to immediately trigger alarms if smoke is detected, so I can evacuate and ensure my safety in case of a fire hazard.	Alarm triggered when smoke is detected.	High	Sprint-1
		USN-3	As an employee, I want the system to provide clear instructions for safe evacuation, so I can follow emergency protocols effectively.	Evacuation instructions provided with alarm	High	Sprint-1
		USN-4	As an employee, I want the system to allow for manual alarm acknowledgment, so I can confirm my safety during evacuation.	Employees can acknowledge alarms through a user interface.	Medium	Sprint-1
		USN-5	As an employee, I want the system to provide historical data for safety drills and compliance audits, so I can ensure regulatory compliance	System stores historical smoke detection data.	Medium	Sprint-2

## 5.2 SOLUTION ARCHITECTURE:



## 6.PROJECT PLANNING AND SCHEDULING:

## 6.1 TECHNICAL ARCHITECTURE:



## 6.2 SPRINT PLANNING AND ESTIMATION:

Jira Software

Your work

Projects

Filters

Dashboards

Teams

Apps

Create

Q Search

🔔

?

⚙️

👤

smoke detection

Software project

You're on the Free plan

UPGRADE

PLANNING

Timeline

Backlog

Board

+ Add view

DEVELOPMENT

Code

Wiki

Add shortcut

Project settings

Projects / smoke detection

Backlog

🔍

👤

👤

👤

👤

Epic 1

Clear filters

Import work

Insights

View settings

Epic

Issues without epic

smoke detection system

Start date: None

Due date: None

View all details

+ Create epic

SD Sprint 1 27 Oct – 2 Nov (3 issues)

0 6 0

Start sprint

IMPLEMENT SMOKE DETECTION SYSTEM WHICH FUNCTIONS EFFECTIVELY AND TRIGGERS A FIRE ALARM WHEN SMOKE IS DETECTED

SD-2 As an employee, I want the system to provide real-time monitoring of the ...

SMOKE DETE...

IN PROGRESS

2

👤

SD-3 As an employee, I want the system to immediately trigger alarms if smoke...

SMOKE DETE...

IN PROGRESS

2

👤

SD-4 As an employee, I want the system to allow for manual alarm acknowledg...

SMOKE DETE...

IN PROGRESS

2

👤

+ Create issue

3 issues | Estimate: 6

SD Sprint 2 3 Nov – 9 Nov (2 issues)

0 6 0

Start sprint

SD-5 As an employee, I want the system to provide clear instructions for safe ev...

SMOKE DETE...

IN PROGRESS

3

👤

SD-6 As an employee, I want the system to provide historical data for safety drill...

SMOKE DETE...

IN PROGRESS

3

👤

## 6.3 SPRINT DELIVERY SCHEDULE:

smoke detection

Software project

You're on the Free plan

UPGRADE

PLANNING

Timeline

Backlog

Board

+ Add view

DEVELOPMENT

Code

Wiki

Add shortcut

Project settings

Projects / smoke detection

Timeline

🔍

👤

👤

👤

👤

Status category

Epic

Give feedback

Share

Export

View settings

	OCT	NOV	DEC	JAN '24
Sprints		SD Sprint 1 SD Sprint 2		
SD-1 smoke detection system				
SD-2 As an emplo... IN PROGRESS				
SD-3 As an emplo... IN PROGRESS				
SD-4 As an emplo... IN PROGRESS				
SD-5 As an emplo... IN PROGRESS				
SD-6 As an emplo... IN PROGRESS				

smoke detection

Software project

You're on the Free plan

UPGRADE

PLANNING

Timeline

Backlog

Board

+ Add view

DEVELOPMENT

Code

Wiki

Add shortcut

Project settings

Projects / smoke detection

All sprints

🔍

👤

👤

👤

👤

Epic

Sprint 2

Clear filters

GROUP BY: None

Import work

Insights

View settings

TO DO

IN PROGRESS 3 OF 3

DONE 2 OF 2

As an employee, I want the system to provide real-time monitoring of the industrial facility's air quality, so I can work in a safe environment.

SMOKE DETECTION SYSTEM

SD-2

2

👤

As an employee, I want the system to provide clear instructions for safe evacuation, so I can follow emergency protocols effectively.

SMOKE DETECTION SYSTEM

SD-5

3

👤

As an employee, I want the system to provide historical data for safety drills and compliance

SMOKE DETECTION SYSTEM

SD-6

3

👤

As an employee, I want the system to immediately trigger alarms if smoke is detected, so I can evacuate and ensure my safety in case of a fire hazard.

SMOKE DETECTION SYSTEM

SD-3

2

👤

As an employee, I want the system to allow for manual alarm acknowledgment, so I can confirm my safety during evacuation.

SMOKE DETECTION SYSTEM

SD-4

2

👤

Quickstart

### 13. 7. CODING & SOLUTIONING (Explain the features added in the project along with code)

**7.1 Collect the dataset 7.2 Importing the libraries 7.3 Read the Dataset 7.4 Data Preparation 7.5 Descriptive statistical 7.6 Visual analysis 7.7 Model Building 7.8 Testing the model 7.9 Performance Testing & Hyperparameter Tuning Activity 7.10 Model Deployment**

#### **7. CODING & SOLUTIONING (Explain the features added in the project along with code) 7.1**

##### **Collect the dataset**

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset

Link: <https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset>

##### **7.2 Importing the libraries**

Import the necessary libraries

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

##### **7.3 Read the Dataset**

Read the dataset with the help of pandas. In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
#Reading the dataset
df = pd.read_csv('/content/smoke_detection_iot.csv')
df.head()
```

	Unnamed: 0	UTC	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	NC0.5	NC1.0	NC2.5	CNT	Fire Alarm
0	0	1654733331	20.000	57.36	0	400	12306	18520	939.735	0.0	0.0	0.0	0.0	0.0	0	0
1	1	1654733332	20.015	56.67	0	400	12345	18651	939.744	0.0	0.0	0.0	0.0	0.0	1	0
2	2	1654733333	20.029	55.96	0	400	12374	18764	939.738	0.0	0.0	0.0	0.0	0.0	2	0
3	3	1654733334	20.044	55.28	0	400	12390	18849	939.736	0.0	0.0	0.0	0.0	0.0	3	0
4	4	1654733335	20.059	54.69	0	400	12403	18921	939.744	0.0	0.0	0.0	0.0	0.0	4	0

##### **7.4 Data Preparation**

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Imbalance data

### 7.4.1 Handling missing values

- Let's find the shape of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used.

```
df.shape
```

```
(62630, 16)
```

```
#Checking for information of features  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 62630 entries, 0 to 62629  
Data columns (total 16 columns):  
#   Column                Non-Null Count  Dtype  
---  ---  
0   Unnamed: 0            62630 non-null  int64  
1   UTC                   62630 non-null  int64  
2   Temperature[C]        62630 non-null  float64  
3   Humidity[%]           62630 non-null  float64  
4   TVOC[ppb]             62630 non-null  int64  
5   eCO2[ppm]             62630 non-null  int64  
6   Raw H2                62630 non-null  int64  
7   Raw Ethanol           62630 non-null  int64  
8   Pressure[hPa]         62630 non-null  float64  
9   PM1.0                 62630 non-null  float64  
10  PM2.5                 62630 non-null  float64  
11  NC0.5                 62630 non-null  float64  
12  NC1.0                 62630 non-null  float64  
13  NC2.5                 62630 non-null  float64  
14  CNT                   62630 non-null  int64  
15  Fire Alarm            62630 non-null  int64  
dtypes: float64(8), int64(8)  
memory usage: 7.6 MB
```

- For checking the null values, `df.isnull()` function is used. To sum those null values we use `.sum()` function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
#Checking for null values
df.isnull().sum()
```

```
Unnamed: 0      0
UTC             0
Temperature[C]   0
Humidity[%]      0
TVOC[ppb]        0
eCO2[ppm]        0
Raw H2           0
Raw Ethanol      0
Pressure[hPa]    0
PM1.0           0
PM2.5           0
NC0.5           0
NC1.0           0
NC2.5           0
CNT             0
Fire Alarm       0
dtype: int64
```

- After dealing with null values, we are removing unnecessary columns as shown below.

```
#Dropping the unnecessary columns
df.drop(columns = ['Unnamed: 0', 'UTC'], axis = 1, inplace = True )
```

#### 7.4.2 Handling Categorical Values

As we can see our dataset has no categorical values. Hence, skipping this step.

#### 7.4.3 Handling Imbalance Data

class imbalance involves dealing with datasets where the classes are not evenly distributed, and one class may be significantly more prevalent than the others. Common techniques for addressing class imbalance include oversampling the minority class, undersampling the majority class, using synthetic data generation techniques such as SMOTE (Synthetic Minority Over-sampling Technique), and using ensemble methods such as bagging and boosting.

```
#Checking the value counts for target column
```

```
df['Fire Alarm'].value_counts()
# 1 - Fire is there
# 0 - No fire
```

```
1    44757
0    17873
Name: Fire Alarm, dtype: int64
```

Therefore, the data set is imbalanced. We are using SMOTE technique to deal with imbalanced dataset after feature selection process.



## 7.5 Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

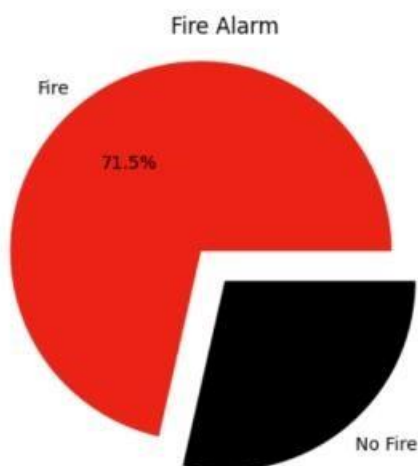
	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	N
count	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.00
mean	15.970424	48.539499	1942.057528	670.021044	12942.453936	19754.257912	938.627649	100.594309	184.467770	491.46
std	14.359576	8.865367	7811.589055	1905.885439	272.464305	609.513156	1.331344	922.524245	1976.305615	4265.66
min	-22.010000	10.740000	0.000000	400.000000	10668.000000	15317.000000	930.852000	0.000000	0.000000	0.00
25%	10.994250	47.530000	130.000000	400.000000	12830.000000	19435.000000	938.700000	1.280000	1.340000	8.82
50%	20.130000	50.150000	981.000000	400.000000	12924.000000	19501.000000	938.816000	1.810000	1.880000	12.45
75%	25.409500	53.240000	1189.000000	438.000000	13109.000000	20078.000000	939.418000	2.090000	2.180000	14.42
max	59.930000	75.200000	60000.000000	60000.000000	13803.000000	21410.000000	939.861000	14333.690000	45432.260000	61482.03

## 7.6 Visual analysis

### 7.6.1 Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot. Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot. graph.

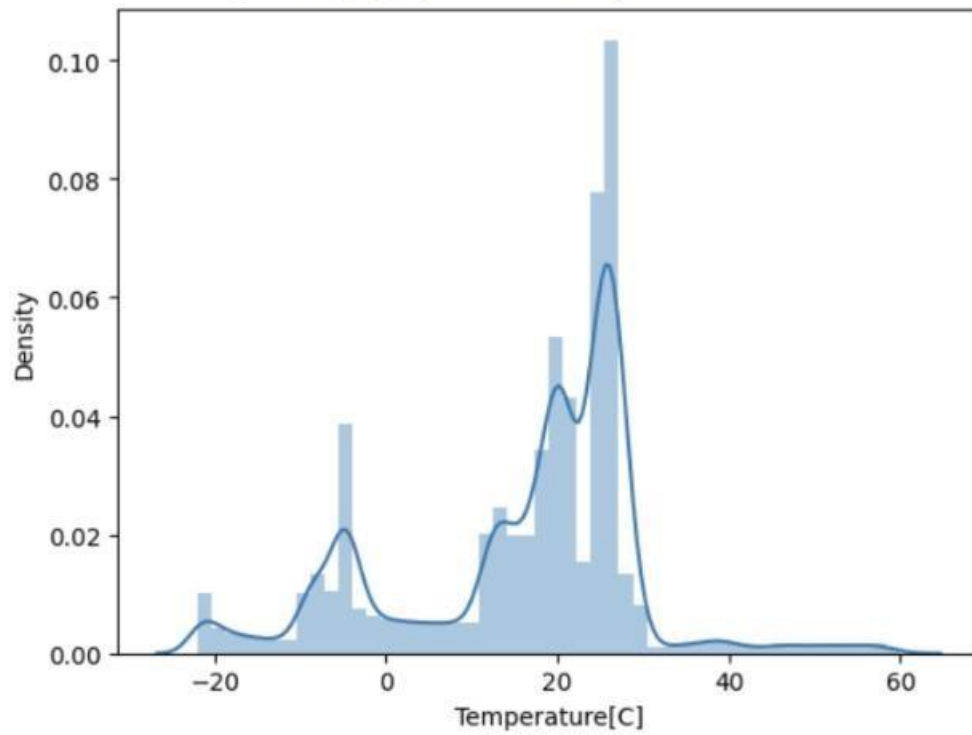
```
plt.pie(df['Fire Alarm'].value_counts(),[0.2,0],labels = ['Fire','No Fire'],autopct='%1.1f%%',colors=['red','black'])  
plt.title('Fire Alarm')  
plt.show()
```



- Univariate analysis of Temperature

```
sns.distplot(df['Temperature[C]'])
```

<Axes: xlabel='Temperature[C]', ylabel='Density'>

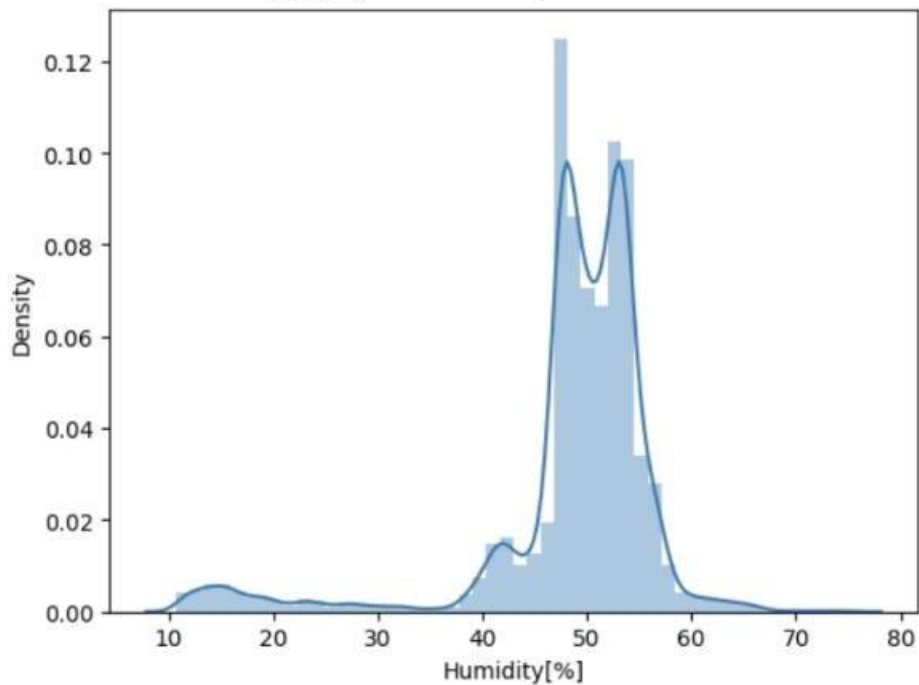


- Analysis of Humidity



```
sns.distplot(df['Humidity[%]'])
```

```
<Axes: xlabel='Humidity[%]', ylabel='Density'>
```



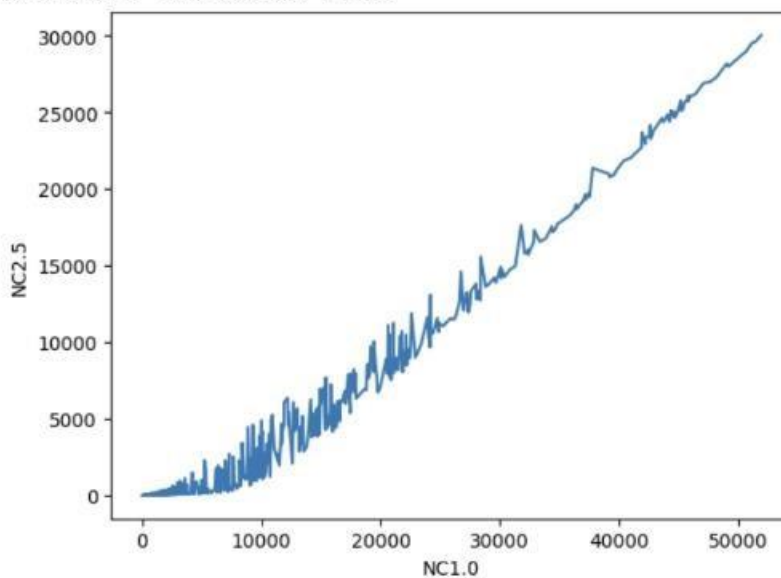
### 7.6.2 Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between

Analysis of variables NC1.0 and NC2.5 using line plot

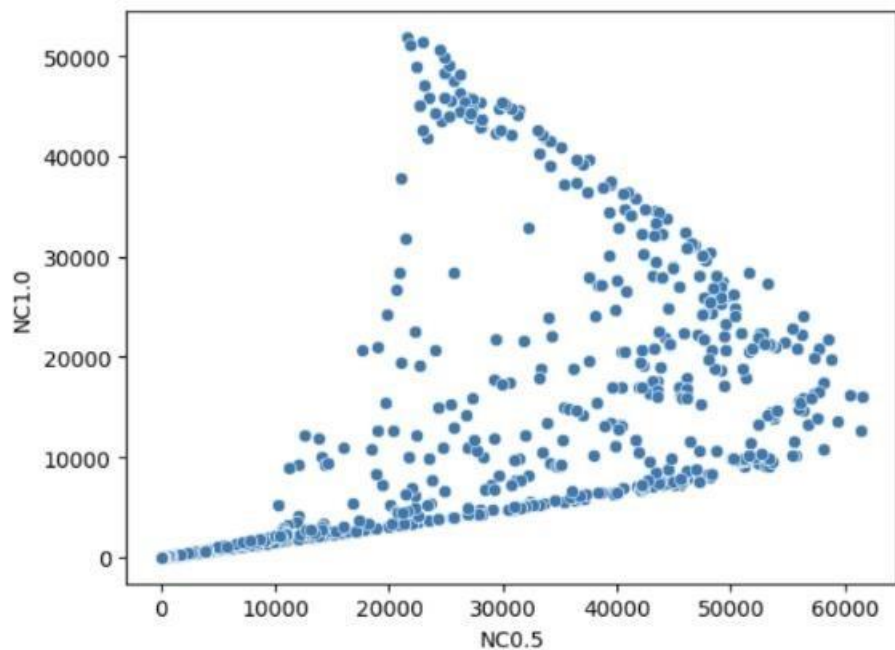
```
import seaborn as sns
sns.lineplot(x='NC1.0',y='NC2.5',data=df)
```

<Axes: xlabel='NC1.0', ylabel='NC2.5'>



- Analysis of NC0.5 and NC1.0 using scatterplot

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(x='NC0.5',y='NC1.0',data=df)
plt.xlabel('NC0.5')
plt.ylabel('NC1.0')
plt.show()
```

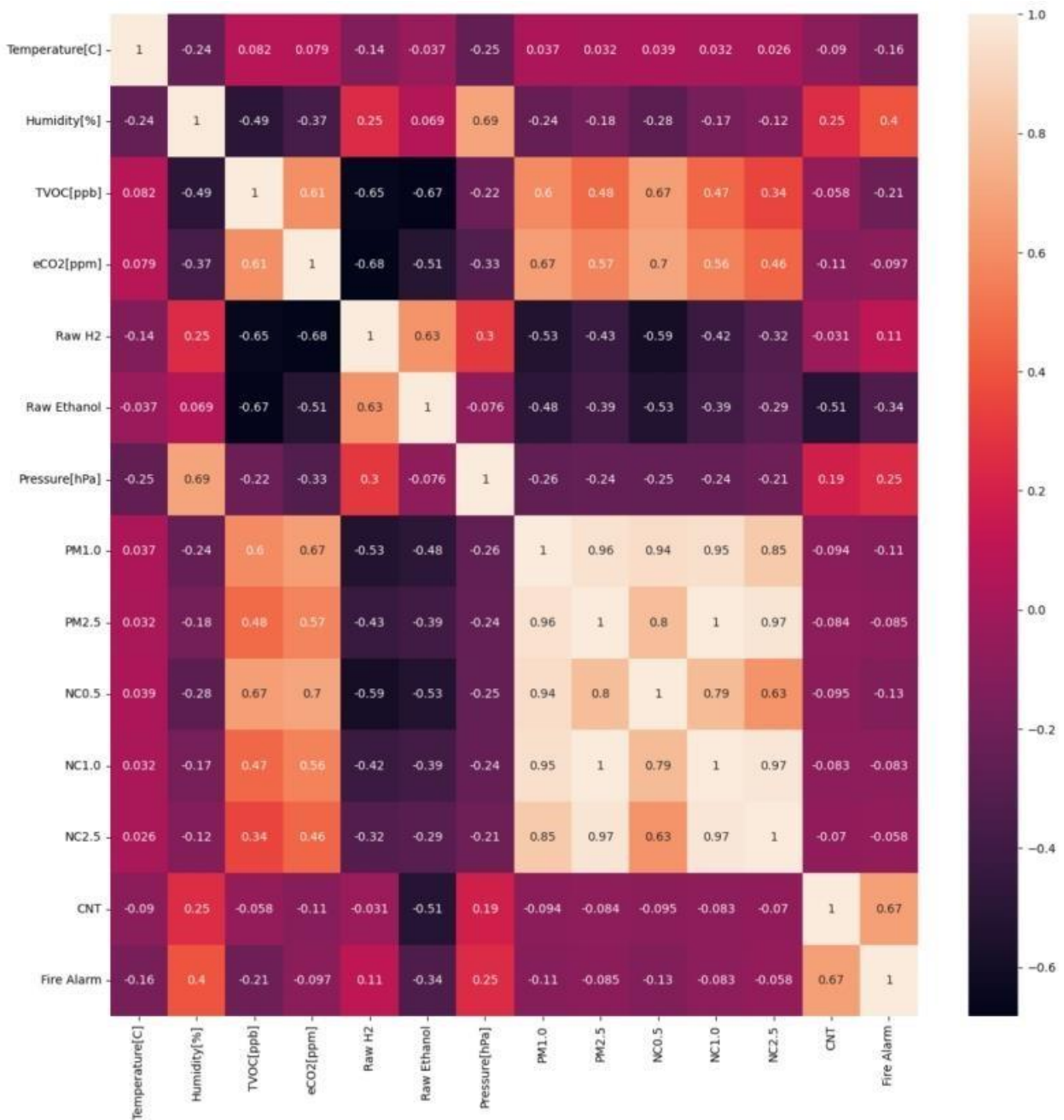


### 7.6.3 Multivariate analysis

multivariate analysis is to find the relation between multiple features

```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr(),annot=True)
```

<Axes: >



Multicollinearity between pairs (NC1.0,NC2.5) and (PM1.0,PM2.5). Therefore, we can drop any one of the columns from a pair

```
df.drop(columns = ['NC1.0','PM1.0'],axis = 1,inplace = True)
```

Finding the correlation between independent variable and dependent variable

```
[52] df.corr()["Fire Alarm"].sort_values(ascending=False)
```

Fire Alarm	1.000000
CNT	0.673762
Humidity[%]	0.399846
Pressure[hPa]	0.249797
Raw H2	0.107007
NC2.5	-0.057707
PM2.5	-0.084916
eCO2[ppm]	-0.097006
NC0.5	-0.128118
Temperature[C]	-0.163902
TVOC[ppb]	-0.214743
Raw Ethanol	-0.340652

Name: Fire Alarm, dtype: float64

REDUCING THE NO.OF FEATURES FOR BETTER MODEL BUILDING

```
[53] df.drop(columns=['NC2.5','PM2.5','eCO2[ppm]'],axis = 1,inplace = True)
```

Out of 11 independent variables, selecting 8 variables which are highly correlated.  
Therefore dropping NC2.5,PM2.5,eCO2 [ppm]

#### DEFINING INDEPENDENT AND DEPENDENT VARIABLES(X,Y)

```
[54] X=df.drop(columns=['Fire Alarm'])  
      y=df['Fire Alarm']
```

```
[55] from sklearn.preprocessing import MinMaxScaler  
      scale = MinMaxScaler()  
      X_scaled = pd.DataFrame(scale.fit_transform(X),columns=X.columns)  
      X_scaled.head()
```

	Temperature[C]	Humidity[%]	TVOC[ppb]	Raw H2	Raw Ethanol	Pressure[hPa]	NC0.5	CNT
0	0.512692	0.723239	0.0	0.522488	0.525685	0.986014	0.0	0.00000
1	0.512875	0.712535	0.0	0.534928	0.547185	0.987013	0.0	0.00004
2	0.513046	0.701520	0.0	0.544179	0.565731	0.986347	0.0	0.00008
3	0.513229	0.690971	0.0	0.549282	0.579682	0.986125	0.0	0.00012
4	0.513412	0.681818	0.0	0.553429	0.591498	0.987013	0.0	0.00016

#### Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

```
[56] from sklearn.model_selection import train_test_split  
      x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=0)
```

Applying Smote technique after feature scaling to avoid imbalance dataset

```
[57] from imblearn.over_sampling import SMOTE  
      smote = SMOTE()  
      x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
```

```
[58] y_train_smote.value_counts()  
  
0      31391  
1      31391  
Name: Fire Alarm, dtype: int64
```

#### 7.7 Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

### 7.7.1 Logistic regression

```
[59] from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score
      from sklearn.metrics import classification_report
      model_lr = LogisticRegression()
      model_lr.fit(x_train_smote,y_train_smote)
      y_pred_test_lr = model_lr.predict(x_test)
      y_pred_train_lr = model_lr.predict(x_train_smote)
      test_acc_lr = accuracy_score(y_test,y_pred_test_lr)
      train_acc_lr =accuracy_score(y_train_smote, y_pred_train_lr)

      print('Logistic Regression Test Accuracy: ',test_acc_lr)
      print(classification_report(y_test,y_pred_test_lr))
```

```
Logistic Regression Test Accuracy: 0.9452871360902656
      precision    recall  f1-score   support

         0         0.85         0.98         0.91         5423
         1         0.99         0.93         0.96        13366

 accuracy                   0.95         18789
 macro avg         0.92         0.96         0.94         18789
 weighted avg        0.95         0.95         0.95         18789
```

The code provided is using the `LogisticRegression` class from the `sklearn.linear_model` module to train a logistic regression model on some data represented by `x_train_smote` and `y_train_smote`, and then making predictions on `x_test` using the trained model. The accuracy of the predictions is evaluated using the `accuracy_score` function from the `sklearn.metrics` module. Finally, the classification report is printed using the `classification_report` function from the same module.

The classification report provides a summary of the model's performance, including metrics such as precision, recall, and F1-score, for each class in the target variable (`y_test`). It gives insights into the model's ability to correctly predict each class, as well as any imbalances or issues with the model's performance. **7.7.2 SVM**



```
[60] from sklearn.svm import SVC
model_svm = SVC()
model_svm.fit(x_train_smote,y_train_smote)
y_pred_test_svm = model_svm.predict(x_test)
y_pred_train_svm = model_svm.predict(x_train_smote)
test_acc_svm = accuracy_score(y_test, y_pred_test_svm)
train_acc_svm = accuracy_score(y_train_smote,y_pred_train_svm)
print('SVM Test Accuracy: ',test_acc_svm)
print(classification_report(y_test,y_pred_test_svm))
```

```
SVM Test Accuracy: 0.9995742189579009
      precision    recall  f1-score   support

     0         1.00      1.00      1.00        5423
     1         1.00      1.00      1.00       13366

 accuracy          1.00
 macro avg         1.00
weighted avg         1.00
```

The SVM classifier is a type of binary classification algorithm that finds the best hyperplane that separates data points of different classes with the maximum margin. The SVC (Support Vector Classifier) from sklearn.svm is used to train the SVM model in the code. Similar to the KNN classifier, the accuracy of the SVM model's predictions is calculated using the accuracy\_score function, and the classification\_report function is used to generate a summary of the model's performance, including precision, recall, and F1-score for each class in the test data.

### 7.7.3 Gradient Boosting

```
[61] from sklearn.ensemble import GradientBoostingClassifier
model_gb = GradientBoostingClassifier()
model_gb.fit(x_train_smote, y_train_smote)
y_pred_test_gb = model_gb.predict(x_test)
y_pred_train_gb = model_gb.predict(x_train_smote)
test_acc_gb = accuracy_score(y_test,y_pred_test_gb)
train_acc_gb = accuracy_score(y_train_smote,y_pred_train_gb)
print('Gradient Boosting Test Accuracy: ',test_acc_gb)
print(classification_report(y_test,y_pred_test_gb))
```

```
Gradient Boosting Test Accuracy: 0.9999467773697376
      precision    recall  f1-score   support

     0         1.00      1.00      1.00        5423
     1         1.00      1.00      1.00       13366

 accuracy          1.00
 macro avg         1.00
weighted avg         1.00
```

The Gradient Boosting classifier is an ensemble learning method that combines multiple weak classifiers to create a stronger, more accurate model. The GradientBoostingClassifier from sklearn.ensemble is used

to train the Gradient Boosting model in the code. Again, the accuracy of the model's predictions is calculated using the `accuracy_score` function, and the `classification_report` function is used to generate a summary of the model's performance, including precision, recall, and F1-score for each class in the test data.

#### 7.7.4 KNN

```
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier()
model_knn.fit(x_train_smote,y_train_smote)
y_pred_test_knn=model_knn.predict(x_test)
y_pred_train_knn=model_knn.predict(x_train_smote)
test_acc_knn=accuracy_score(y_test,y_pred_test_knn)
train_acc_knn=accuracy_score(y_train_smote,y_pred_train_knn)
print('KNN Test Accuracy: ',test_acc_knn)
```

KNN Test Accuracy: 0.9999467773697376

The KNN classifier is a type of instance-based learning that classifies new data points based on the class labels of their k-nearest neighbors in the training data. The `KNeighborsClassifier` from `sklearn.neighbors` is used to train the KNN model in the code. The accuracy of the model's predictions is calculated using the `accuracy_score` function, and the `classification_report` function is used to generate a summary of the model's performance, including precision, recall, and F1-score for each class in the test data.

#### 7.8 Testing the model

Here we have tested with Gradient Boosting algorithm. You can test with all algorithm. With the help of `predict()` function.

```
[63] result = model_lr.predict([[20.05,55.28,0,12390,18849,939.736,0,3]])
      print(result)
```

[0]

#### 7.9 Performance Testing & Hyperparameter Tuning Activity

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

##### 7.9.1 Compare the model

Comparing all the Models as shown.



```

import pandas as pd
from sklearn.metrics import accuracy_score, classification_report
model_names = ['Logistic Regression', 'SVM', 'Gradient Boosting', 'KNN']
y_pred_tests = [ y_pred_test_lr, y_pred_test_svm, y_pred_test_gb, y_pred_test_knn]
results_df = pd.DataFrame(columns=['Model', 'Test accuracy', 'Precision', 'Recall', 'F1-score'])
for i, model_name in enumerate(model_names):
    model = model_names[i]
    y_pred_test = y_pred_tests[i]

    test_acc = accuracy_score(y_test, y_pred_test)
    classification = classification_report(y_test, y_pred_test, output_dict=True)
    precision = classification['macro avg']['precision']
    recall = classification['macro avg']['recall']
    f1_score = classification['macro avg']['f1-score']

    results_df = results_df.append({'Model' : model_name,
                                   'Test Accuracy' : test_acc,
                                   'Precision' : precision,
                                   'Recall' : recall,
                                   'F1-score' : f1_score}, ignore_index=True)

print(results_df)

```

	Model	Test accuracy	Precision	Recall	F1-score	\
0	Logistic Regression	NaN	0.921939	0.955572	0.936074	
	Test Accuracy					
0		0.945287				
	Model	Test accuracy	Precision	Recall	F1-score	\
0	Logistic Regression	NaN	0.921939	0.955572	0.936074	
1	SVM	NaN	0.999646	0.999317	0.999481	
	Test Accuracy					
0		0.945287				
1		0.999574				
	Model	Test accuracy	Precision	Recall	F1-score	\
0	Logistic Regression	NaN	0.921939	0.955572	0.936074	
1	SVM	NaN	0.999646	0.999317	0.999481	
2	Gradient Boosting	NaN	0.999908	0.999963	0.999935	
	Test Accuracy					
0		0.945287				
1		0.999574				
2		0.999947				
	Model	Test accuracy	Precision	Recall	F1-score	\
0	Logistic Regression	NaN	0.921939	0.955572	0.936074	
1	SVM	NaN	0.999646	0.999317	0.999481	
2	Gradient Boosting	NaN	0.999908	0.999963	0.999935	
3	KNN	NaN	0.999908	0.999963	0.999935	
	Test Accuracy					
0		0.945287				
1		0.999574				
2		0.999947				
3		0.999947				

After calling the function, the results of models are displayed as output. Hence svm, knn, gradient boosting seems to be overfitting. Considering logistic regression model as appropriate model From all the models, we considered logistic Regression to avoid over-fitting problem.

## 7.9.2

```

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
confusion = confusion_matrix(y_test, y_pred_test_lr)
accuracy = accuracy_score(y_test, y_pred_test_lr)
classification_rep = classification_report(y_test, y_pred_test_lr)

print("Confusion Matrix:\n", confusion)
print(f"Accuracy Score: {accuracy}")
print("Classification Report:\n", classification_rep)

```

```

Confusion Matrix:
[[ 5314  109]
 [  919 12447]]
Accuracy Score: 0.9452871360902656
Classification Report:

```

	precision	recall	f1-score	support
0	0.85	0.98	0.91	5423
1	0.99	0.93	0.96	13366
accuracy			0.95	18789
macro avg	0.92	0.96	0.94	18789
weighted avg	0.95	0.95	0.95	18789

Using confusion\_matrix, accuracy\_score, classification\_report functions to evaluate the performance of your logistic regression model. After running this code, the confusion matrix, accuracy score, and a classification report is displayed, which will help to assess how well the model is performing on the test data.

## 7.10 Model Deployment

### 7.10.1 Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```

import pickle
with open('smoke.pkl', 'wb') as file:
    pickle.dump(model_lr, file)

```

### 7.10.2 Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the users where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

### 7.10.3 Integrate with Web Framework

For this project create two HTML files namely

- home.html
- predict.html

- submit.html

#### 7.10.4 Build Python code

```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (`__name__`) as argument

```
5 with open('smoke.pkl', 'rb') as file:
6     model = pickle.load(file)
7
8 app = Flask(__name__)
9
10 @app.route('/')
```

Render HTML page:

```
11 def home():
12     return render_template('home.html')
13
14 @app.route('/predict')
15 def predict():
16     return render_template('predict.html')
17
18 @app.route('/submit', methods=['POST'])
19 def submit():
20     temperature = float(request.form['temperature'])
21     humidity = float(request.form['humidity'])
22     tvoc = float(request.form['tvoc'])
23     raw_h2 = float(request.form['raw_h2'])
24     raw_ethanol = float(request.form['raw_ethanol'])
25     pressure = float(request.form['pressure'])
26     nc0_5 = float(request.form['nc0_5'])
27     cnt = float(request.form['cnt'])
28     final_features = np.array([temperature, humidity, tvoc, raw_h2, raw_ethanol, pressure, nc0_5, cnt])
29     final_features = final_features.reshape(1, -1)
30     prediction = model.predict(final_features)[0]
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier. In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method. Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

```

31
32     if prediction == 0:
33         prediction_text = 'The input does not indicate smoke detection.'
34     else:
35         prediction_text = 'The input indicates smoke detection.'
36     return render_template('submit.html', prediction_text=prediction_text)
37
38 if __name__ == '__main__':
39     app.run(debug=True)

```

### 7.10.5 Run the web application

Open the .py file in VSCode Click on run the python code in the terminal

Access the localhost url in the terminal. Example url looks like (http://127.0.0.1:5000)

## 8.PERFORMANCE TESTING

### 8.1 PERFORMANCE METRICS

S.No.	Parameter	Values	Screenshot																														
1.	Metrics	<b>Classification Model (Logistic Regression) :</b> Confusion Matrix - , Accuracy Score- & Classification Report -	 <pre>from sklearn.metrics import confusion_matrix, accuracy_score, classification_report confusion = confusion_matrix(y_test, y_pred_test_lr) accuracy = accuracy_score(y_test, y_pred_test_lr) classification_rep = classification_report(y_test, y_pred_test_lr)  print("Confusion Matrix:\n", confusion) print(f"Accuracy Score: {accuracy}") print("Classification Report:\n", classification_rep)</pre> <p>Confusion Matrix:</p> <pre>[[ 5314  109]  [  925 12441]]</pre> <p>Accuracy Score: 0.9449678003086912</p> <p>Classification Report:</p> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.85</td><td>0.98</td><td>0.91</td><td>5423</td></tr><tr><td>1</td><td>0.99</td><td>0.93</td><td>0.96</td><td>13366</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.94</td><td>18789</td></tr><tr><td>macro avg</td><td>0.92</td><td>0.96</td><td>0.94</td><td>18789</td></tr><tr><td>weighted avg</td><td>0.95</td><td>0.94</td><td>0.95</td><td>18789</td></tr></tbody></table>		precision	recall	f1-score	support	0	0.85	0.98	0.91	5423	1	0.99	0.93	0.96	13366	accuracy			0.94	18789	macro avg	0.92	0.96	0.94	18789	weighted avg	0.95	0.94	0.95	18789
	precision	recall	f1-score	support																													
0	0.85	0.98	0.91	5423																													
1	0.99	0.93	0.96	13366																													
accuracy			0.94	18789																													
macro avg	0.92	0.96	0.94	18789																													
weighted avg	0.95	0.94	0.95	18789																													

## 9.RESULTS

### 9.1 OUTPUT SCREENSHOTS

## Smoke Detector Prediction

*Welcome to our website dedicated to proactively predicting smoke hazards using machine learning.*

*Our state-of-the-art smoke detectors are designed to enhance safety in homes and businesses by accurately detecting smoke early on.*

Predict

Home

### Smoke Detection Prediction

Temperature[C]:

Humidity[%]:

TVOC[ppb]:

Raw H2:

Raw Ethanol:

Pressure[hPa]:

NC0.5:

CNT:

Submit





[Home](#)

### Smoke Detection Prediction

Temperature[C]:  
150

Humidity[%]:  
48.51

TVOC[ppb]:  
1120

Raw H2:  
67

Raw Ethanol:  
35

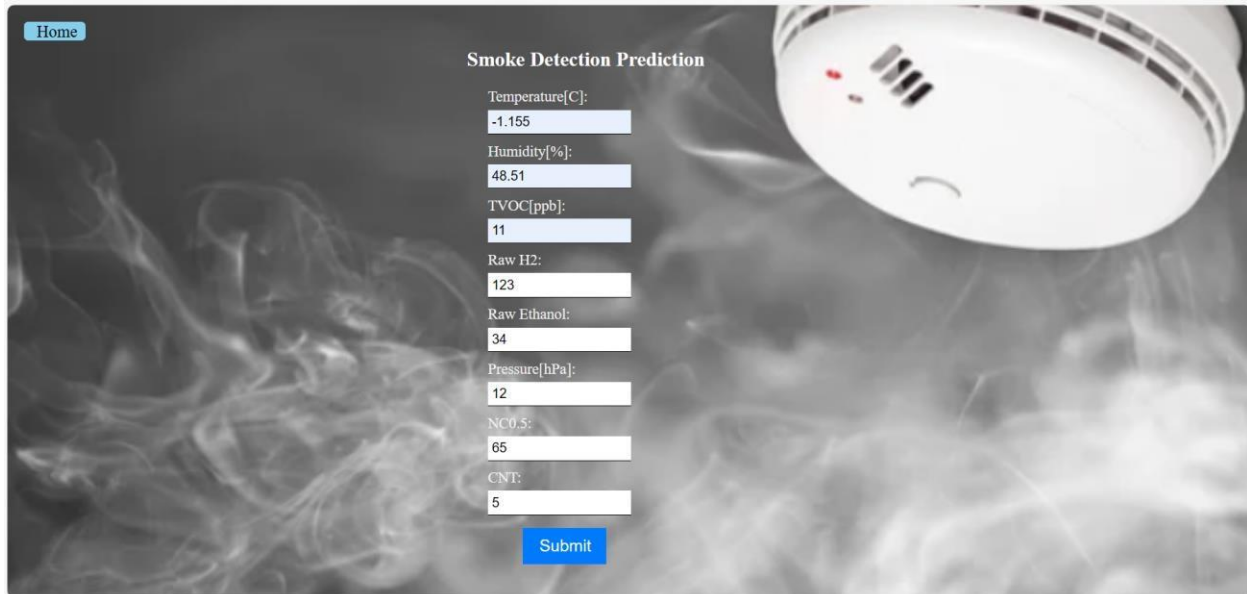
Pressure[hPa]:  
55

NC0.5:  
13.91

CNT:  
4064

Submit



The interface features a dark background with a swirling smoke pattern on the left and a white smoke detector on the right. A central form titled "Smoke Detection Prediction" contains several input fields with pre-filled values. A "Home" button is in the top left, and a "Submit" button is at the bottom right of the form.

Home

### Smoke Detection Prediction

Temperature[C]:  
-1.155

Humidity[%]:  
48.51

TVOC[ppb]:  
11

Raw H2:  
123

Raw Ethanol:  
34

Pressure[hPa]:  
12

NC0.5:  
65

CNT:  
5

Submit



## 10.ADVANTAGES AND DISADVANTAGES:

### ADVANTAGES:

**Early Detection of Smoke:** The system provides early detection of smoke, enabling prompt action to prevent potential fire accidents and minimize damage to assets.

**Real-time Monitoring:** The system offers real-time monitoring of smoke levels, allowing for timely alerts and responses to potential fire hazards.

**Scalability and Expandability:** The system can accommodate additional smoke sensors or devices as per the requirements of the industrial setting, allowing for scalability and expandability.

**Integration with Existing Systems:** The system can integrate with existing fire safety systems, such as sprinklers or fire suppression systems, to enhance overall fire safety measures.

**Detailed Reporting and Analysis:** The system analyzes the collected data to provide detailed reports for further analysis and decision-making.

#### **DISADVANTAGES:**

**Reliability on Internet Connectivity:** The system heavily relies on stable internet connectivity for data transmission and remote monitoring. Any disruptions in the internet connection can impact the system's functionality and responsiveness.

**Limited Compatibility:** The system's compatibility with existing infrastructure or fire safety systems in industries may be limited. Integration with other systems or devices may require additional customization or technical adjustments.

**Privacy Concerns:** The use of IoT devices for smoke detection raises privacy concerns as data is collected and transmitted. It is crucial to ensure that proper data privacy measures are in place to protect sensitive information.

**Training and User Familiarity:** Proper training and user familiarity are essential for effective utilization of the system. Employees or personnel responsible for monitoring and managing the system may require training to ensure its optimal use.

**Maintenance and Upkeep:** Regular maintenance and upkeep of the system are necessary to ensure its proper functioning. This includes sensor calibration, software updates, and battery replacements, which can add to the overall maintenance costs.

#### **11.CONCLUSION:**

In conclusion, the implementation of smoke detection using IoT and triggering an alarm in industries offers significant benefits in terms of fire safety and risk reduction. By leveraging IoT technology, this system provides real-time monitoring, early smoke detection, and prompt alarm triggering, ensuring timely response to potential fire hazards. The integration of highly sensitive smoke sensors and wireless connectivity enables efficient data transmission and analysis, enhancing the accuracy and reliability of the system. Additionally, the remote monitoring and control capabilities provide convenience and accessibility for users to monitor the system status and receive real-time notifications from anywhere. The scalability and integration options allow for seamless integration with existing fire safety systems, further enhancing overall safety measures. With compliance to safety standards, this system provides a reliable and efficient solution for smoke detection in industries, mitigating the risks associated with fire accidents and ensuring the well-being of employees and protection of assets. Overall, the implementation of smoke detection using IoT and triggering an alarm in industries is a crucial step towards enhancing fire safety measures and reducing the potential impact of fire incidents in industrial settings.



**12.FUTURE SCOPE:** The future scope for smoke detection using IoT and triggering an alarm in industries includes advancements in sensor technology, integration with building management systems, and the implementation of artificial intelligence and machine learning algorithms. Additionally, advanced analytics and predictive maintenance, integration with emergency response systems, and smart evacuation systems are potential areas for development. Energy efficiency measures, integration with the broader IoT ecosystem, and cloud-based analytics and storage solutions can further enhance the capabilities of the system. Continuous research and development efforts will drive innovation and address emerging challenges in industrial fire safety.

**13.APPENDIX:**

**SOURCE CODE:**

<https://drive.google.com/file/d/1TEXCK98yWgKPK7N-L7KygnMTRQxRpemJ/view?usp=sharing>

**PROJECT DEMO LINK:**

<https://drive.google.com/file/d/1CVnLAgoGNGxh0MCuaZxQf0BKDMwY04P4/view?usp=sharing>

**GITHUB:**

<https://github.com/smartinternz02/SI-GuidedProject-596513-1697363041.git>