```
# Check total number of rows and columns
```

```
(150, 9)
```

Fig. 8. Total rows and colum

The Fig.8. provides an overview of the dataset's structure, showing the total number of data entries (rows) and the various attributes (columns) recorded. Understanding the dataset size is crucial in validating that there is enough data to train and test the model effectively. A well-sized dataset helps ensure reliable classification of toxic gas levels.

```
safe_df = df[df['Status'] == 'Safe']
safe_df
```

	Timestamp	Sensor ID	CO (ppm)	NO_2 (ppb)	SO_2 (ppb)	O_3 (ppb)	H_2S (ppb)	NH_3 (ppb)	Status
89	2024-04-09 12:30:00	GAS003	5.69	42	15	30	1	22	Safe
119	2024-04-12 12:30:00	GAS002	2.73	31	7	50	3	12	Safe
121	2024-04-13 08:30:00	GAS002	4.66	58	5	67	4	20	Safe
135	2024-04-14 10:30:00	GAS001	5.71	55	6	59	4	26	Safe
143	2024-04-15 09:30:00	GAS002	1.80	52	5	30	6	34	Safe
145	2024-04-15 10:30:00	GAS003	5.44	28	13	66	2	30	Safe

Fig. 9. Output of safe state

The Fig.9. displays filtered records from the dataset that are categorised as "Safe" based on predefined thresholds. These gas readings indicate that the environment is within acceptable and non-hazardous limits. This output assures that the detection system correctly identifies areas or timeframes where no immediate health risks are present.

```
warning_df = df[df['Status'] == 'Warning']
warning_df
```

	Timestamp	Sensor ID	CO (ppm)	NO_2 (ppb)	SO_2 (ppb)	O_3 (ppb)	H_2S (ppb)	NH_3 (ppb)	Status
0	2024-04-01 08:00:00	GAS002	2.22	67	16	69	2	96	Warning
2	2024-04-01 09:00:00	GAS001	3.58	70	6	61	1	33	Warning
8	2024-04-01 12:00:00	GAS003	9.37	33	12	31	7	37	Warning
11	2024-04-02 08:30:00	GAS002	8.37	80	7	24	5	57	Warning
12	2024-04-02 09:00:00	GAS003	3.48	24	7	62	4	91	Warning
13	2024-04-02 09:30:00	GAS002	5.62	43	7	35	3	91	Warning
15	2024-04-02 10:30:00	GAS001	2.93	71	12	39	1	78	Warning
29	2024-04-03 12:30:00	GAS001	6.81	81	8	31	5	57	Warning
34	2024-04-04 10:00:00	GAS003	6.89	31	16	36	2	98	Warning
36	2024-04-04 11:00:00	GAS001	5.56	47	16	67	2	73	Warning
42	2024-04-05 09:00:00	GAS001	9.34	22	9	40	2	26	Warning
55	2024-04-06 10:30:00	GAS001	5.68	39	10	52	1	73	Warning
57	2024-04-06 11:30:00	GAS002	1.82	64	11	27	7	67	Warning
58	2024-04-06 12:00:00	GAS002	10.40	58	7	61	6	33	Warning
75	2024-04-08 10:30:00	GAS003	9.12	55	11	41	5	76	Warning
76	2024-04-08 11:00:00	GAS001	7.52	82	7	47	6	74	Warning
78	2024-04-08 12:00:00	GAS002	10.32	72	14	62	2	87	Warning
79	2024-04-08 12:30:00	GAS002	1.25	39	15	18	8	76	Warning

Fig. 10. Output of warning state

This portion of the output highlights gas readings that fall into the "Warning" category. These values are not critically dangerous but are high enough to warrant caution. The model flags these readings to help anticipate future risks, enabling timely preventive action and enhancing safety monitoring.

```
danger_df = df[df['Status'] == 'Dangerous']
danger_df
```

	Timestamp	Sensor ID	CO (ppm)	NO_2 (ppb)	SO_2 (ppb)	O_3 (ppb)	H_2S (ppb)	NH_3 (ppb)	Status
1	2024-04-01 08:30:00	GAS002	7.45	118	20	27	2	38	Dangerous
3	2024-04-01 09:30:00	GAS002	2.58	49	16	29	7	142	Dangerous
4	2024-04-01 10:00:00	GAS001	6.52	61	30	49	4	79	Dangerous
5	2024-04-01 10:30:00	GAS003	16.53	109	5	35	1	35	Dangerous
6	2024-04-01 11:00:00	GAS003	11.30	115	5	28	9	96	Dangerous
7	2024-04-01 11:30:00	GAS001	13.88	52	21	40	10	128	Dangerous
9	2024-04-01 12:30:00	GAS001	11.11	71	21	59	4	125	Dangerous
10	2024-04-02 08:00:00	GAS002	18.12	120	16	34	1	31	Dangerous
14	2024-04-02 10:00:00	GAS003	19.14	45	17	20	3	42	Dangerous
16	2024-04-02 11:00:00	GAS003	17.51	101	6	59	10	99	Dangerous
17	2024-04-02 11:30:00	GAS003	14.07	43	13	15	8	127	Dangerous
18	2024-04-02 12:00:00	GAS003	16.08	99	15	64	10	104	Dangerous
19	2024-04-02 12:30:00	GAS003	1.38	49	20	66	9	111	Dangerous
20	2024-04-03 08:00:00	GAS003	13.52	99	9	19	3	78	Dangerous
21	2024-04-03 08:30:00	GAS002	4.28	100	6	45	8	102	Dangerous
22	2024-04-03 09:00:00	GAS003	11.20	115	25	70	10	85	Dangerous
23	2024-04-03 09:30:00	GAS003	18.94	96	18	61	3	39	Dangerous
24	2024-04-03 10:00:00	GAS003	17.04	77	25	69	1	117	Dangerous

Fig. 11. Output of dangerous state

The dangerous state output pinpoints the most severe gas readings—those that surpass hazardous limits. These entries reflect critical danger and would typically trigger immediate alerts to the users. The classification helps in identifying unsafe conditions that may require evacuation or emergency intervention.

```
df['Status'].value_counts()
```

Status	count
Dangerous	116
Warning	28
Safe	6

Fig. 12. Total count of each state

This summarized count displays how many entries belong to each category—Safe, Warning, and Dangerous. It gives a big-picture view of how balanced or skewed the dataset is across safety levels, and provides insight into the overall environmental condition being analyzed.

This comparative output ranks the three states (Safe, Warning, Dangerous) from least to most critical. This allows stakeholders to quickly identify whether most conditions are manageable or if dangerous gas levels dominate the dataset. It supports decisions regarding the urgency and extent of safety measures required in Fig.13.

```python
# Display all together
display(safest)
display(most_dangerous)
display(biggest_warning)
```

	Timestamp	Sensor ID	CO (ppm)	NO_2 (ppb)	SO_2 (ppb)	O_3 (ppb)	H_2S (ppb)	NH_3 (ppb)	Status
89	2024-04-09 12:30:00	GAS003	5.69	42	15	30	1	22	Safe

	Timestamp	Sensor ID	CO (ppm)	NO_2 (ppb)	SO_2 (ppb)	O_3 (ppb)	H_2S (ppb)	NH_3 (ppb)	Status
74	2024-04-08 10:00:00	GAS003	19.71	23	12	23	2	118	Dangerous

	Timestamp	Sensor ID	CO (ppm)	NO_2 (ppb)	SO_2 (ppb)	O_3 (ppb)	H_2S (ppb)	NH_3 (ppb)	Status
123	2024-04-13 09:30:00	GAS001	6.09	85	15	18	6	37	Warning

Fig. 13. Safest, medium and the most dangerous state

This bar graph provides a visual representation of the frequency of each safety level across the dataset. The clear, comparative height of each bar makes it easy to interpret which gas level occurs most often, helping to identify patterns and inform preventive strategies.

```python
# Plotting
plt.figure(figsize=(6,4))
level_counts.plot(kind='bar', color=['red', 'orange', 'green'])
plt.title('Gas Danger Level Distribution')
plt.xlabel('Level')
plt.ylabel('Number of Records')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

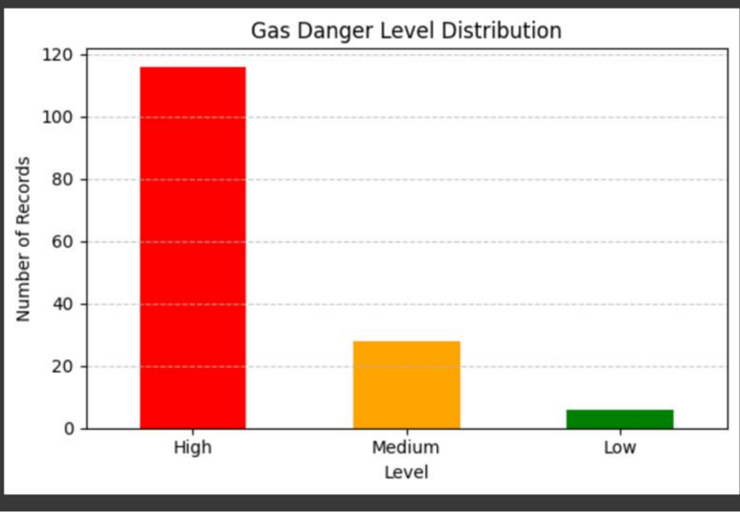

Fig. 14. Gas danger level distribution in Bar graph

The pie chart illustrates the proportional distribution of each gas safety state. This format is especially useful for showing the share of each category at a glance, helping users quickly understand whether the environment being monitored is mostly safe or hazardous.

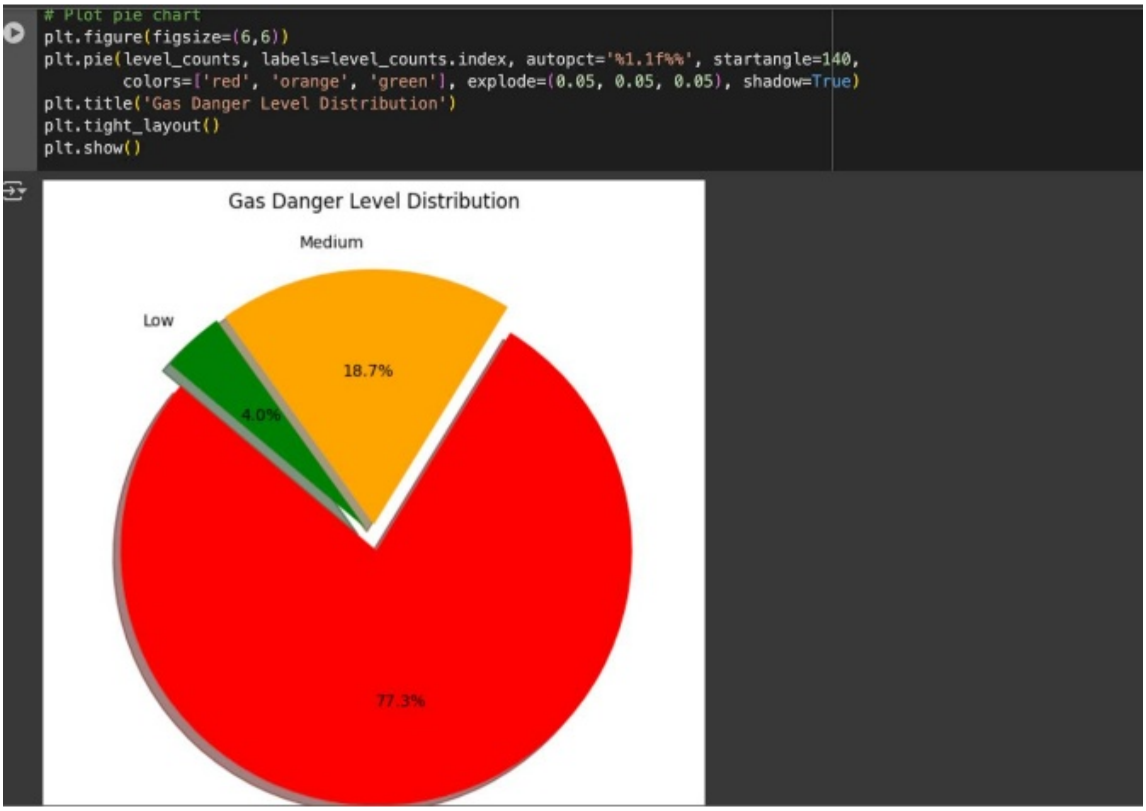

Fig. 15. Gas Danger Level Distribution in Pie chart

This bar graph provides a visual representation of the frequency of each safety level across the dataset. The clear, comparative height of each bar makes it easy to interpret which gas level occurs most often, helping to identify patterns and inform preventive strategies.

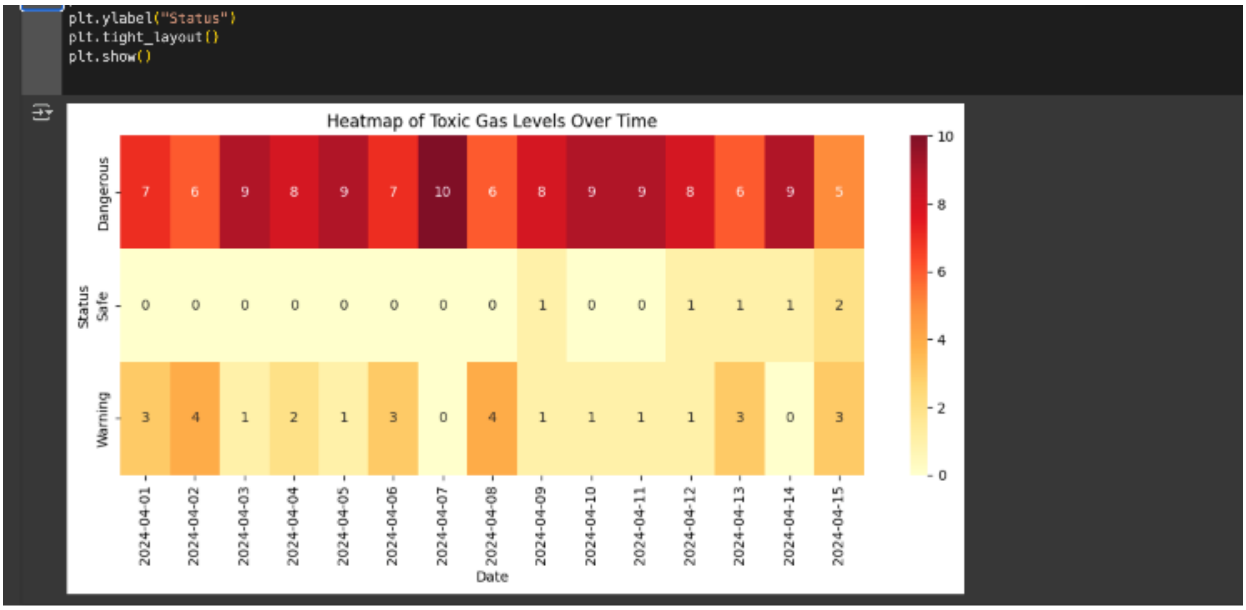

Fig. 16. Gas Distribution over a time period in heatmap

This line chart tracks the progression of gas danger levels over time. It captures rises and falls in gas readings, making it easier to spot cyclical patterns or anomalies. This chart is crucial for long-term monitoring and predicting future spikes in toxicity.

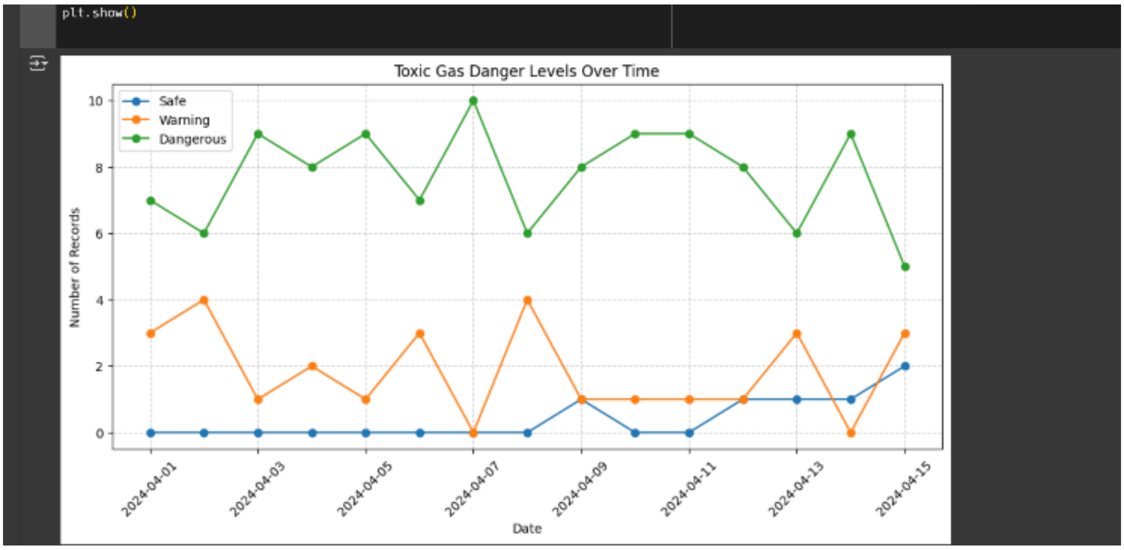

Fig. 17. Gas level distribution in lie chart

The weapon detection system uses a trained Convolutional Neural Network (CNN) to identify unauthorized military weapons from images. The system was developed using a dataset of defense-related objects and deployed using Google Colab. When a suspicious object such as a gun or explosive device is detected in an uploaded image, the model sends a real-time alert to the user via email along with a live camera link. This automatic detection process eliminates the need for manual monitoring and enhances security by providing immediate awareness of potential threats. This solution is especially useful in border surveillance, military zones, and public safety operations.

Fig. 18. Armed man with weapon

This image shows the successful prediction output of the machine learning model. A gun has been detected in the image, clearly labeled by a bounding box along with a prediction confidence score. The object is correctly recognized as a weapon, showcasing that the model is functioning as intended for identifying firearms.

The first image again displays a gun, detected similarly with a bounding box and class label. This confirms consistency in detection performance across different gun images, even with variations in background and orientation.

Fig. 19.Armed guns

The Fig.20. shows a military aircraft. If this was part of a defense object detection dataset, the model may be identifying or logging it for surveillance purposes. However, it's unclear if this was classified as an unauthorized object. If the aircraft was labeled or boxed, it may indicate the model's capability to detect large-scale military equipment.

Fig. 20.Aircrafts

The Fig.21. shows a test image containing an object that visually resembles an explosive device (e.g., a time bomb or wired device). The system detects the object and classifies it accordingly. This confirms that the model has been trained to identify not just hand-held weapons but also suspicious explosive components.

Fig. 21.Bombs

The Fig.22. displays missiles being detected in the frame. The machine learning model identifies them, demonstrating its broader scope to detect advanced military-grade weaponry. The system can recognize missile shapes and flag them as threats in relevant surveillance scenarios.

Fig. 22.missiles

The Fig.23. shows that the email notification automatically sent by the system upon detection of an unauthorized object such as a weapon. The email includes evidence such as the detected image and the time of occurrence, ensuring authorities receive timely and actionable alerts to prevent potential threats in restricted areas.

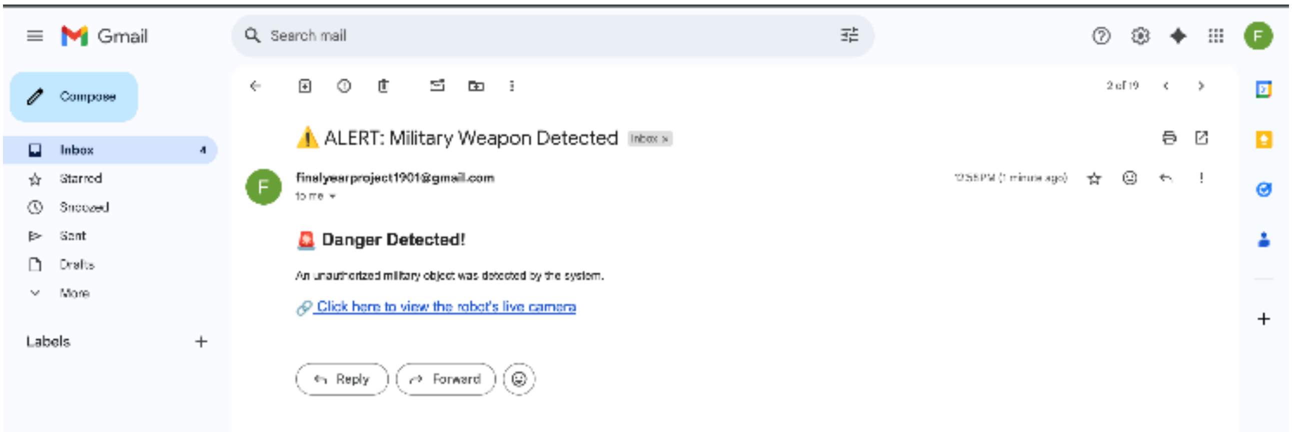

Fig. 23. E-mail Alert when detected an unauthorised object

The Fig.24. shows that the ESP32 CAM is connected with devices such as phone or laptops like that. This is how the vision process is done. When this the operator can manually see via this and if they saw any bomb they can diffuse via gripper also we implemented the ml in this so it can also automatically detect when if it see any bomb and sends the email which it it detects.

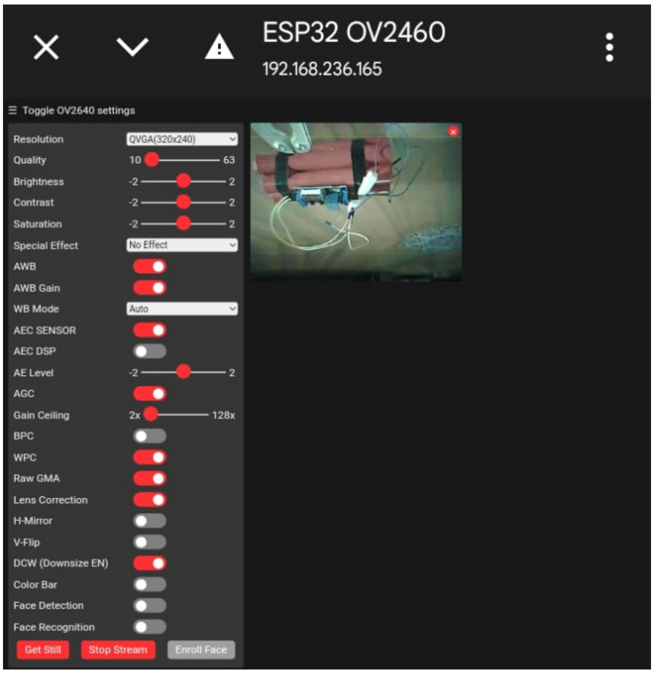

Fig. 24. ESP32 cam live feed

The Fig.25. shows that when a metal detector detects landmine it will send the exact location of the bomb.