24EEE431 - AI & Edge Computing

SMART DAMAGE/ LEAKAGE DETECTION SYSTEM

Batch 03

Aakash Sriram - CB.EN.U4ELC22001

Mohithaa K - CB.EN.U4ELC22029

Mukhil R - CB.EN.U4ELC22033

Pavitra A - CB.EN.U4ELC22040

INTRODUCTION

Pipelines are essential for fuel and gas distribution in urban and industrial areas. Leakages can cause catastrophic consequences like explosions, fires, and toxic exposure.

The aim is to develop and deploy a machine learning-based gas detection system using the Random Forest classifier on a Raspberry Pi, capable of classifying gas types into four categories — No Gas, Perfume, Mixture, and Smoke and triggering an alarm system when Smoke is detected, ensuring targeted and timely hazard alerts.

A mobile app to alert users in real-time, display the type of gas detected, and provide a 'Shut-off' control to remotely stop the gas flow instantly.

PROBLEM STATEMENT

Smart cities frequently use communal LPG pipeline networks requiring continuous monitoring.

Undetected gas leaks can lead to:

- Severe accidents and health hazards.
- Financial losses due to damage and service disruption.

Current leak detection methods:

- Depend heavily on manual inspections.
- Utilize outdated, inefficient monitoring systems.
- Are slow, reactive, and unable to provide real-time alerts.

NEED

A smart system capable of real-time gas leakage detection, and remote control to enhance safety and operational efficiency of gas pipelines.

SOLUTION

The solution proposes a smart, automated, and remote system that:

- Detects leaks,
- Classifies gas types,
- Triggers alerts before critical thresholds are breached,
- Allows remote shutdown of gas supply from a mobile platform.

ABOUT THE DATASET

Dataset Size:

- Total Records: 6400
- Features: 9 (S.No + 7 sensor readings + Gas Category)

Sensors Used:

- MQ2, MQ3, MQ5, MQ6, MQ7, MQ8, MQ135

Target Classes (Gas Categories):

- NoGas, Perfume, Smoke, Mixture

Structure:

- Sensor readings captured numerically (integer values).

	Serial Number	MQ2	MQ3	MQ5	MQ6	MQ7	MQ8	MQ135	Gas
0	0	555	515	377	338	666	451	416	NoGas
1	1	555	516	377	339	666	451	416	NoGas
2	2	556	517	376	337	666	451	416	NoGas
3	3	556	516	376	336	665	451	416	NoGas
4	4	556	516	376	337	665	451	416	NoGas

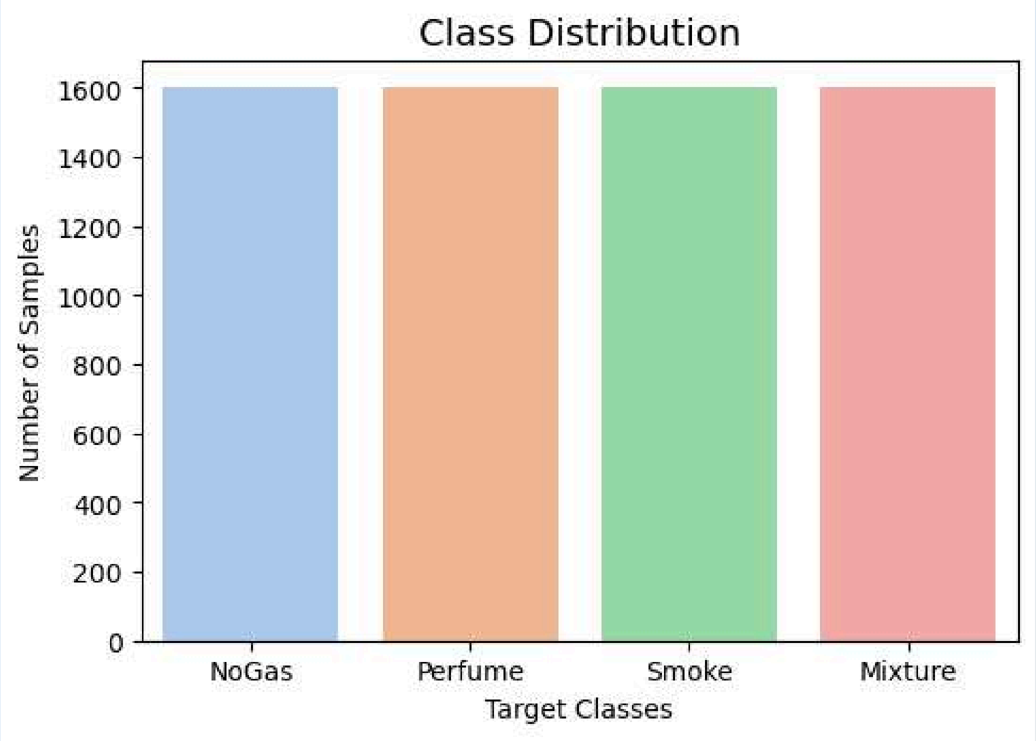
	Serial Number	MQ2	MQ3	MQ5	MQ6	MQ7	MQ8	MQ135
count	6400.000000	6400.000000	6400.000000	6400.000000	6400.000000	6400.000000	6400.000000	6400.000000
mean	799.500000	677.593438	462.024688	404.579063	399.758750	565.952031	542.473750	416.727031
std	461.916214	92.913955	70.284038	55.672249	45.091353	83.133693	151.020217	76.681407
min	0.000000	502.000000	337.000000	291.000000	311.000000	361.000000	220.000000	275.000000
25%	399.750000	591.000000	405.000000	366.000000	366.000000	524.000000	447.000000	354.000000
50%	799.500000	701.000000	486.000000	400.000000	393.000000	576.000000	576.000000	437.000000
75%	1199.250000	756.000000	529.000000	443.000000	426.000000	629.000000	642.000000	473.000000
max	1599.000000	824.000000	543.000000	596.000000	524.000000	796.000000	794.000000	589.000000

EDA

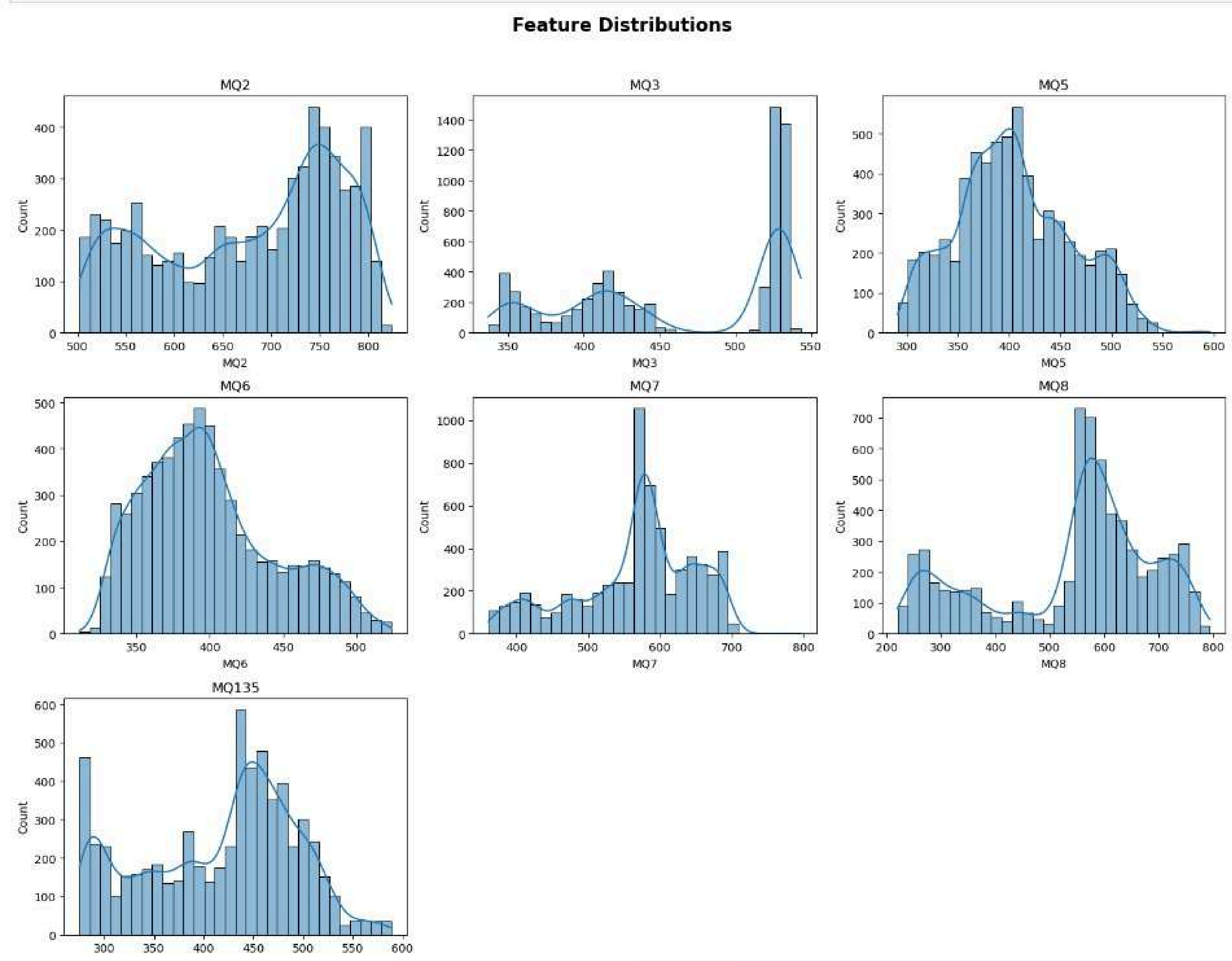
Null Values Check

```
Missing Values:
 MQ2       0
MQ3       0
MQ5       0
MQ6       0
MQ7       0
MQ8       0
MQ135     0
Gas       0
dtype: int64
```

Class Balance

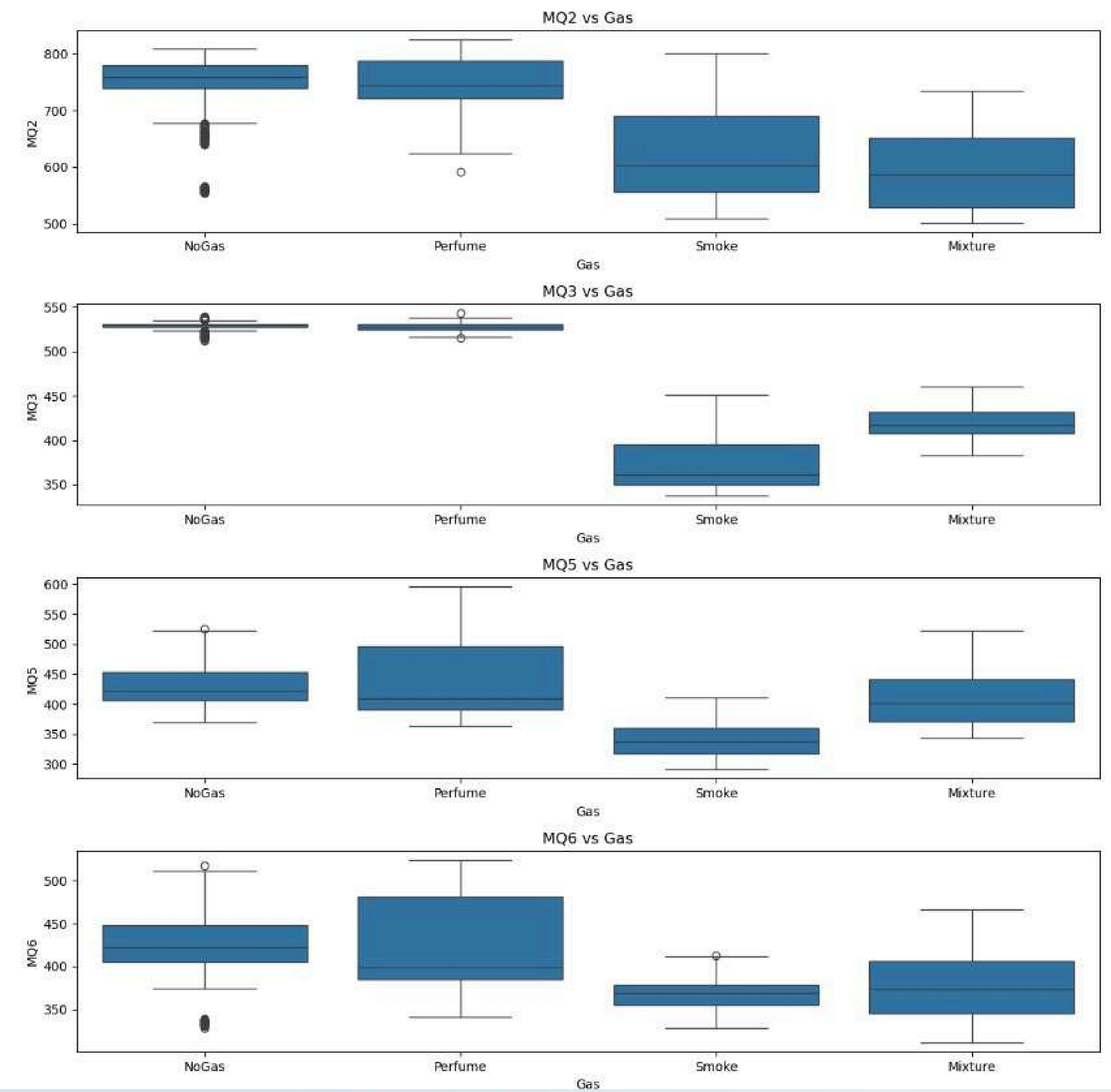


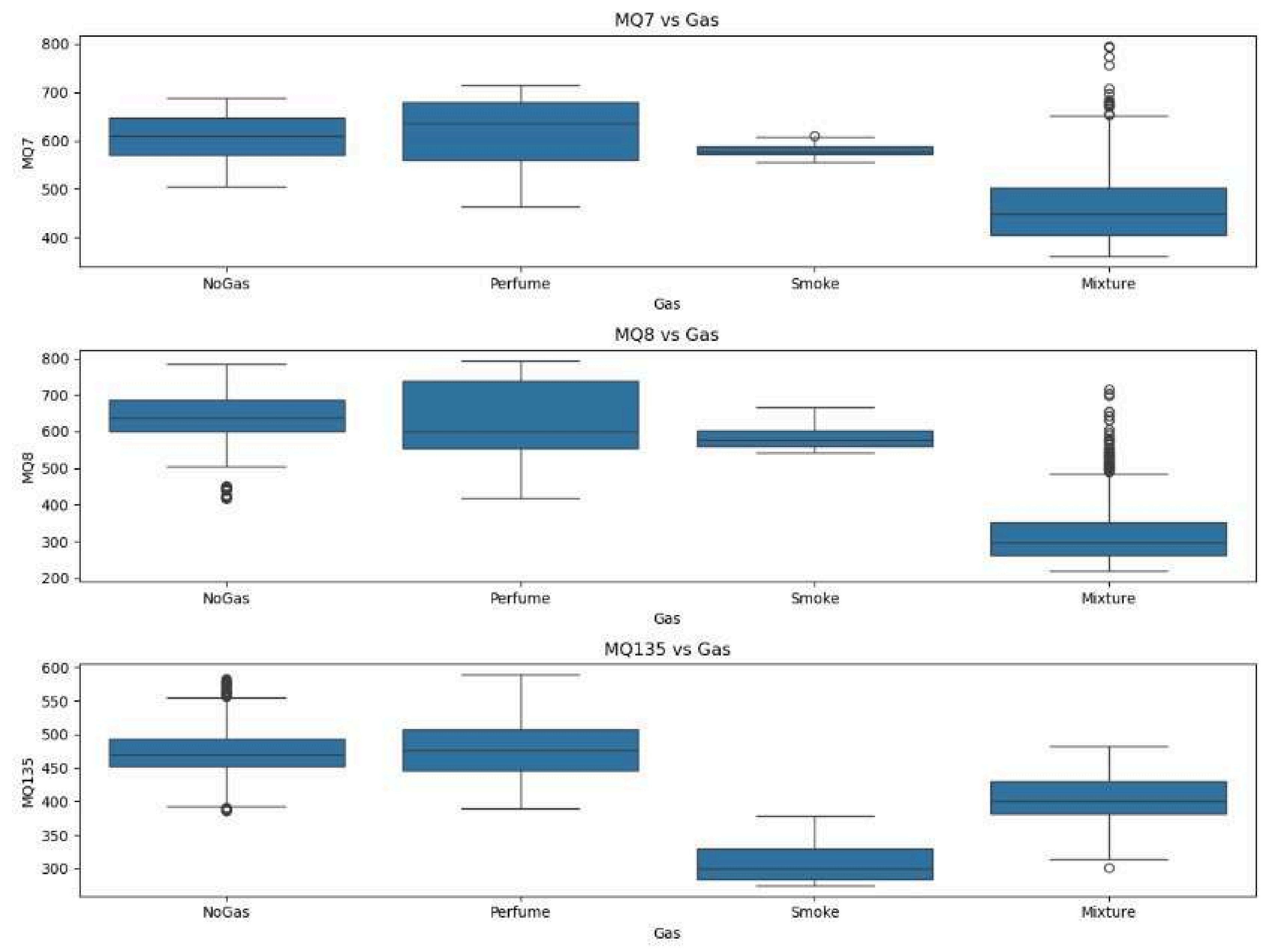
Feature Distribution



EDA

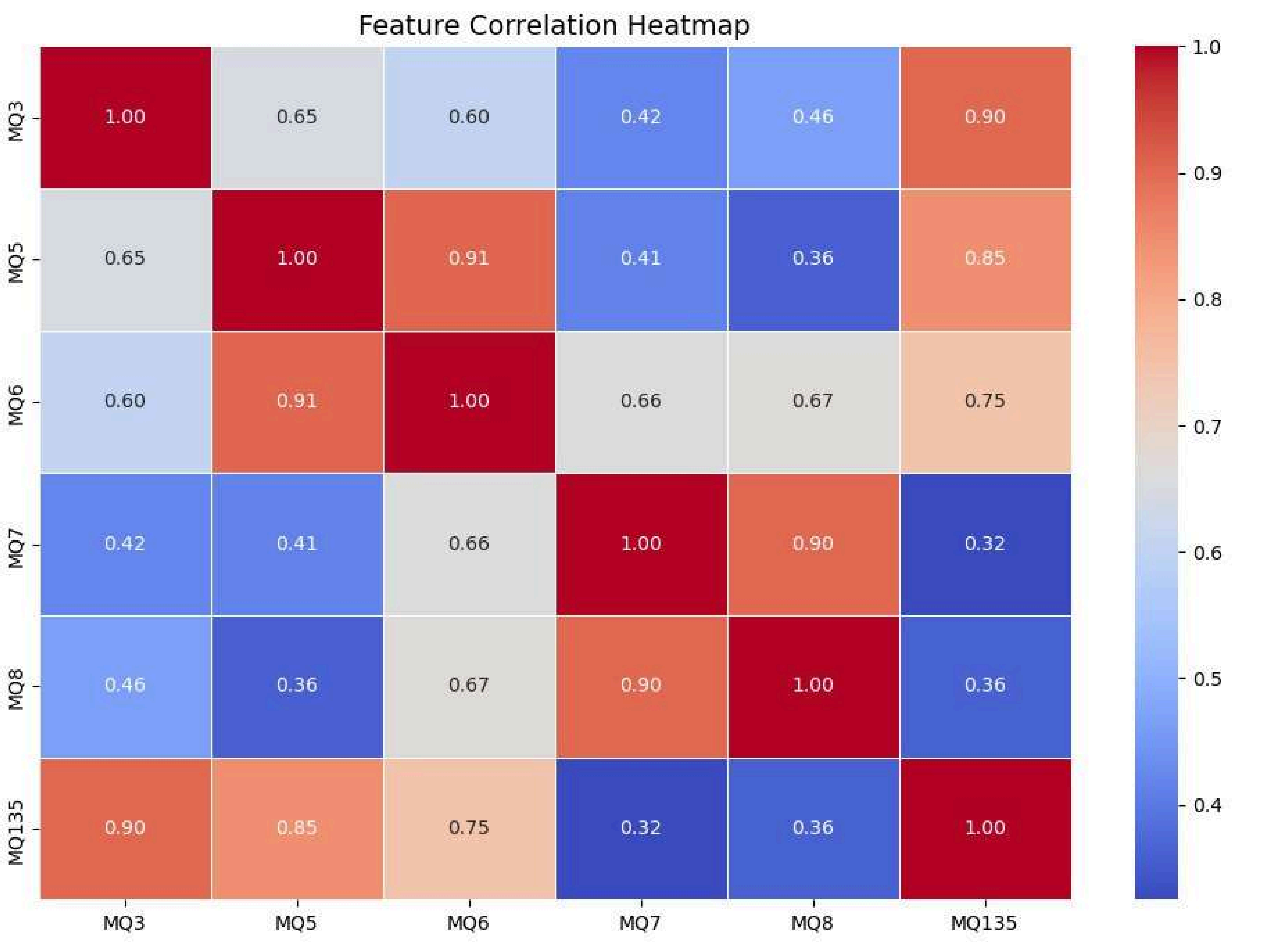
Outlier Detection





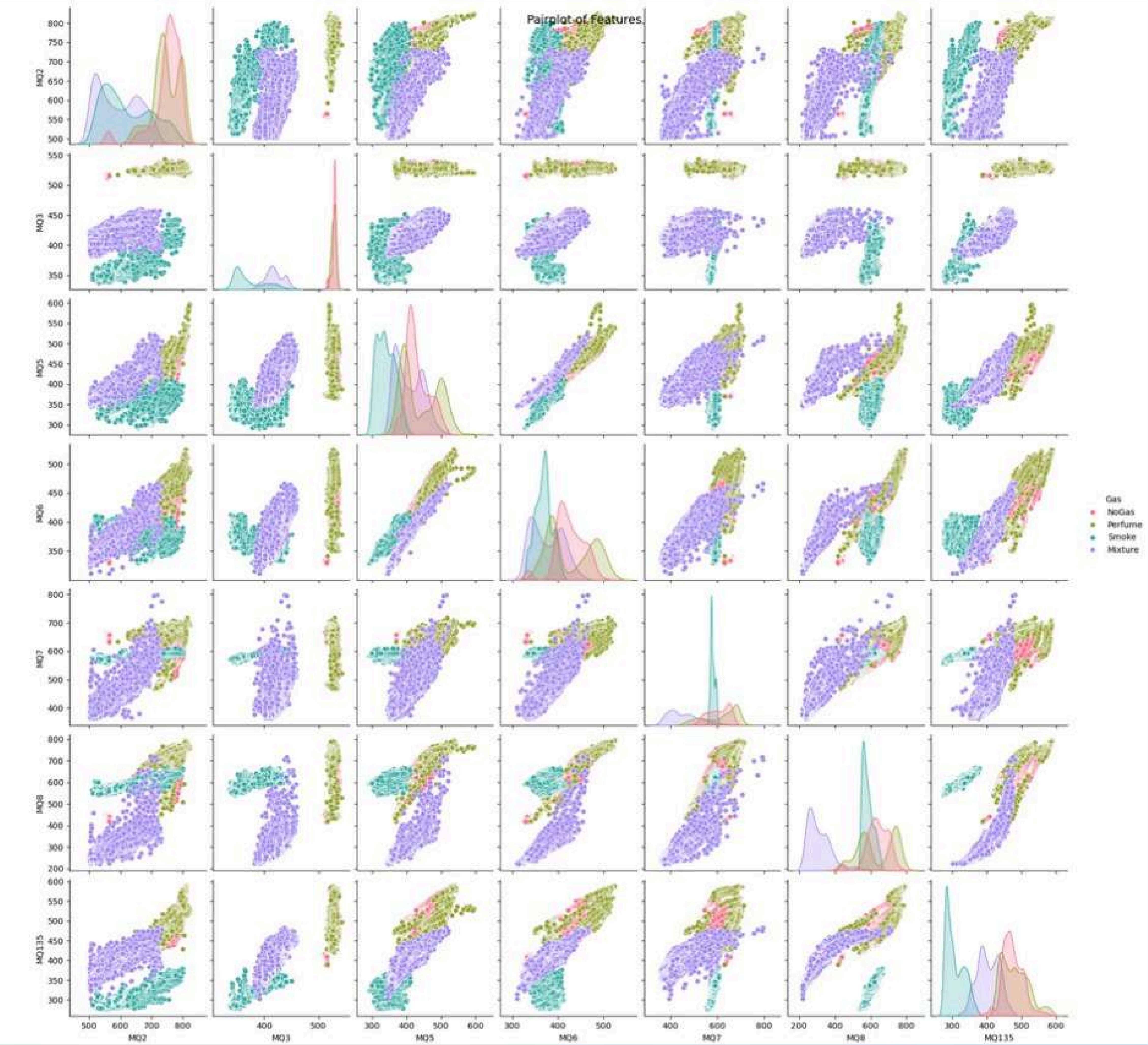
EDA

Correlation Matrix



EDA

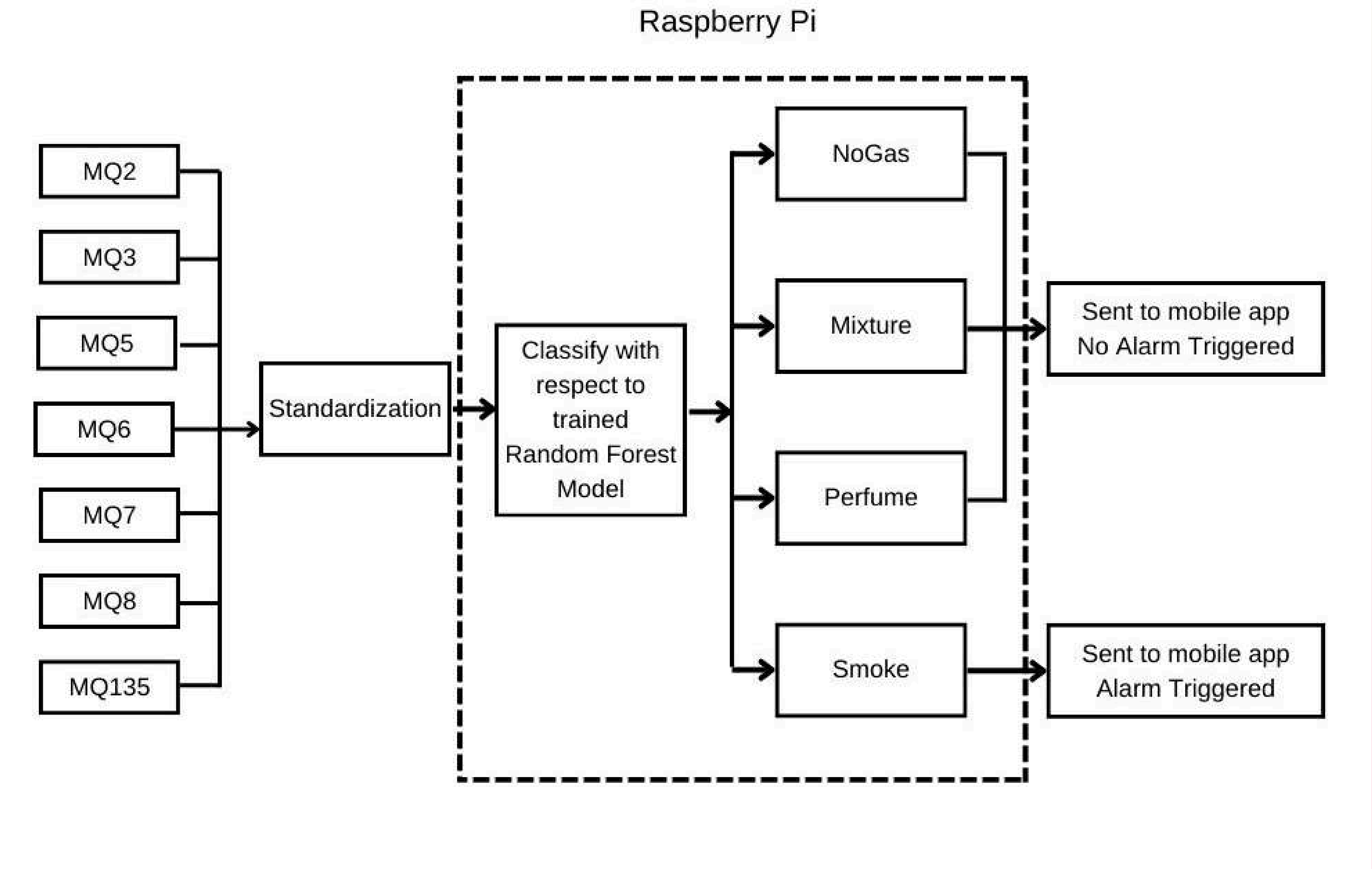
Feature Relationships



METHODOLOGY

Now, let's look at how the system works.

BLOCK DIAGRAM



DATA COLLECTION & PREPROCESSING

- The dataset is collected from gas sensors measuring various environmental parameters.
- Unnecessary columns are removed to ensure relevant features are used for classification.
- Features are normalized using StandardScaler (Z-score normalization) to improve model accuracy.
- The target variable (gas type) is encoded using LabelEncoder for machine learning processing.

```python
file_path = "Dataset.csv"
df = pd.read_csv(file_path)

# Drop the first column (assuming it's an index or unwanted column)
df = df.iloc[:, 1:]

# Separate features and target variable
X = df.iloc[:, :-1]  # Features
y = df.iloc[:, -1]   # Target variable

# Encode target labels (if they are categorical)
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Save the LabelEncoder for later use
joblib.dump(label_encoder, "label_encoder.pkl")
print("[INFO] LabelEncoder saved as 'label_encoder.pkl'")

# Apply StandardScaler (Z-score normalization)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Save the StandardScaler
joblib.dump(scaler, "scaler.pkl")
print("[INFO] StandardScaler saved as 'scaler.pkl'")
```

MODEL DEVELOPMENT & TRAINING

- A Random Forest Classifier is used for classification due to its robustness in handling complex data.
- Hyperparameter tuning is performed using GridSearchCV to optimize performance.
- The trained model, along with the scaler and label encoder, is saved for deployment.

```python
# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.2, random_state=42)

# Define parameter grid for hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Grid Search with 5-fold Cross Validation
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

# Best model after tuning
best_rf = grid_search.best_estimator_
print(f"\nBest Parameters: {grid_search.best_params_}")

# Save the trained model
joblib.dump(best_rf, "random_forest_model.pkl")
print("[INFO] RandomForest model saved as 'random_forest_model.pkl'")
```

REAL-TIME SMOKE DETECTION

- The system receives sensor data via a socket connection from the client device.
- The incoming data is preprocessed using the saved StandardScaler.
- The trained Random Forest model predicts the type of gas detected.
- The detected gas type is sent to a mobile application via MQTT for real-time monitoring.

```
(aiedge) pi@raspberrypi:~/Downloads/aiedge $ python3 client.py
[CLIENT] Connected to server.
[CLIENT] Sending normal sensor values...
[CLIENT] Sent data: 748,339,323,473,466,578,428
[CLIENT] Sent data: 622,453,409,529,579,792,359
[CLIENT] Sent data: 509,474,559,494,783,506,373
[CLIENT] Sent data: 560,505,391,346,652,540,322
[CLIENT] Sent data: 754,407,498,443,525,719,518
```

```
[SERVER] Detected: Smoke
[SERVER] Detected: Perfume
[SERVER] Detected: Mixture
[SERVER] Detected: Smoke
```

ALERT MECHANISM & CONTROL

- If smoke is detected, a timer starts to monitor its persistence.
- If smoke persists for 9 seconds, an alarm is triggered, and a warning is sent via MQTT.
- A RESET command is required to resume normal operation.

```
[CLIENT] Sent SMOKE data: 538,407,388,401,599,591,297
[CLIENT] Sent SMOKE data: 781,413,375,395,573,595,278
[CLIENT] Sent SMOKE data: 628,415,363,378,580,540,282
[CLIENT] Sent SMOKE data: 725,425,317,357,593,667,357
[CLIENT] Sent SMOKE data: 665,423,380,388,610,645,288
[CLIENT] STOP command received. Halting pipeline.
[CLIENT] Pipeline Closed. Waiting for RESET...
```

```
[SERVER] Detected: Smoke
[SERVER] Smoke detected, monitoring duration...
[SERVER] Detected: Smoke
[SERVER] Detected: Smoke
[SERVER] Detected: Smoke
[SERVER] Detected: Smoke
[SERVER] Smoke persisted for 9s! Triggering alarm...
[SERVER] Received MQTT message: ENABLE
[SERVER] Waiting for RESET command...
[SERVER] Waiting for RESET command...
[SERVER] Waiting for RESET command...
[SERVER] Waiting for RESET command...
[SERVER] Waiting for RESET command...
[SERVER] Waiting for RESET command...
[SERVER] Waiting for RESET command...
```

ALERT MECHANISM & CONTROL

Gas Detected

Detected Gas: Mixture
smoke/status 29-Mar 10:16:55

Detected Gas: Mixture
smoke/status 29-Mar 10:16:53

Detected Gas: Perfume
smoke/status 29-Mar 10:16:51

Detected Gas: Smoke
smoke/status 29-Mar 10:16:36

Detected Gas: Smoke
smoke/status 29-Mar 10:16:33

Detected Gas: Smoke
smoke/status 29-Mar 10:16:30

Detected Gas: Smoke
smoke/status 29-Mar 10:16:27

Detected Gas: Mixture
smoke/status 29-Mar 10:16:25

Detected Gas: Mixture
smoke/status 29-Mar 10:16:23

Detected Gas: Perfume
smoke/status 29-Mar 10:16:21

Alarm Detected

Smoke confirmed! Alarm triggered.
smoke/alert 29-Mar 10:16:36

Smoke confirmed! Alarm triggered.
smoke/alert 29-Mar 10:15:48

Smoke confirmed! Alarm triggered.
smoke/alert 29-Mar 10:15:11

Smoke confirmed! Alarm triggered.
smoke/alert 29-Mar 10:06:23

ALARM BUZZER

↑10:16:51

SOLUTION APPROACH

Now, let's explore the key features of the solution.

IoT-Based Smoke Detection

Gas sensors continuously monitor air quality, and Raspberry Pi processes real-time data using a Random Forest Classifier.

Real-Time Alerts

If smoke is detected for 9 seconds, an alarm is triggered, and an MQTT-based alert is sent to the mobile app.

Web-Based Monitoring

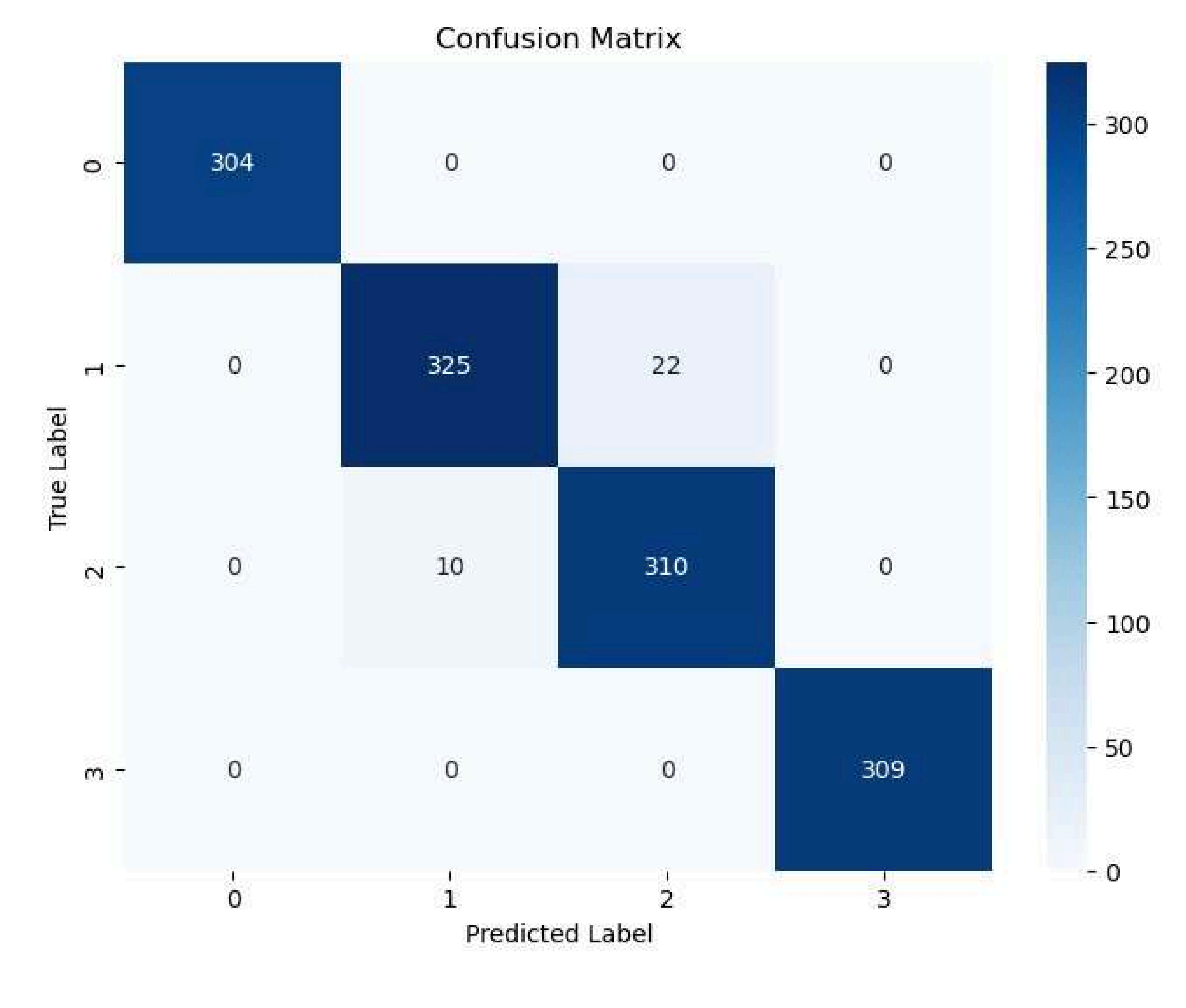
A web dashboard displays real-time data, logs, and system status with a RESET option.

Integration with Emergency Response Systems

Alerts are automatically sent to emergency teams for quick action.

RESULT

Confusion Matrix



Classification Report

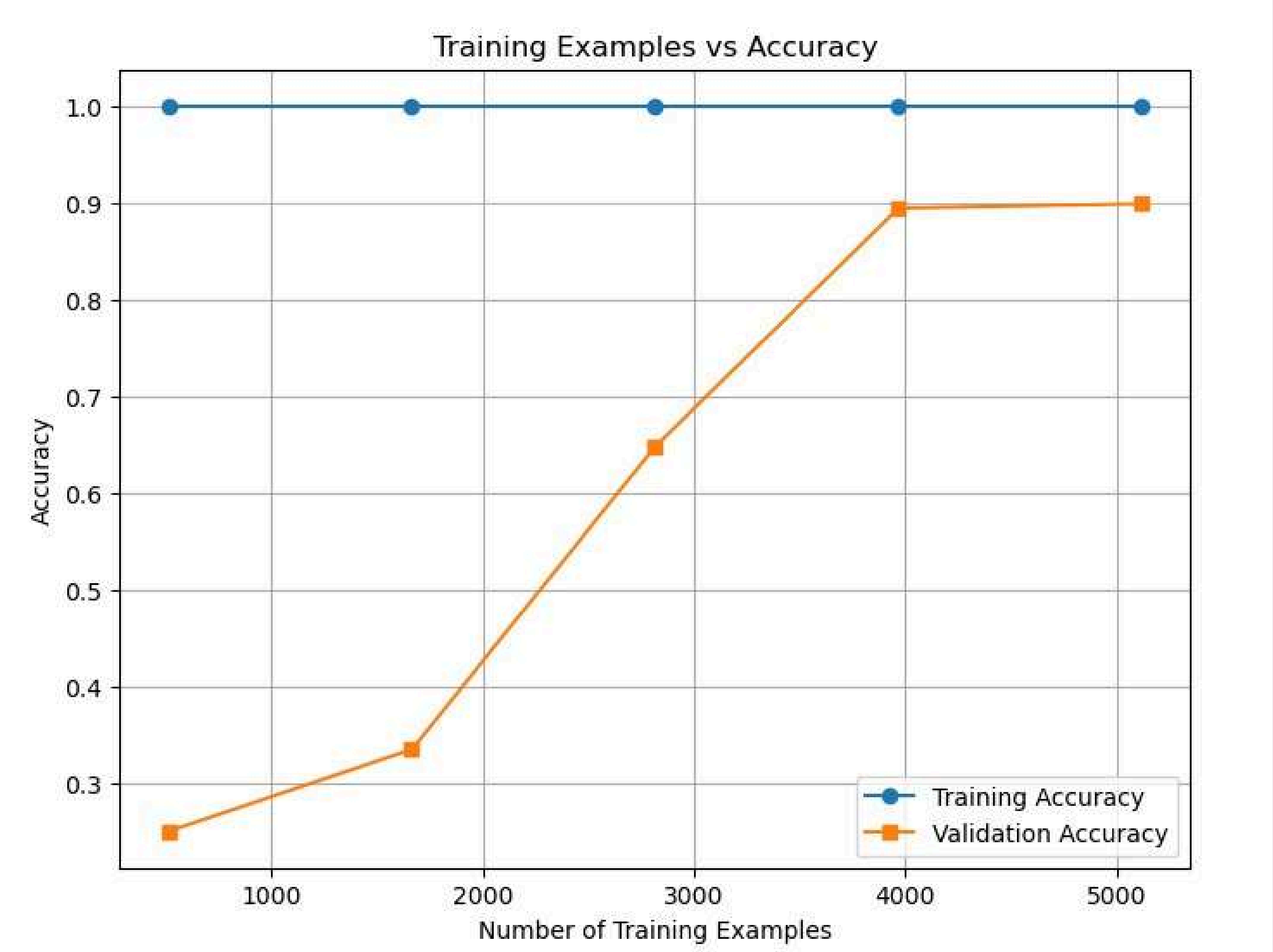
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	304
1	0.97	0.94	0.95	347
2	0.93	0.97	0.95	320
3	1.00	1.00	1.00	309
accuracy			0.97	1280
macro avg	0.98	0.98	0.98	1280
weighted avg	0.98	0.97	0.98	1280

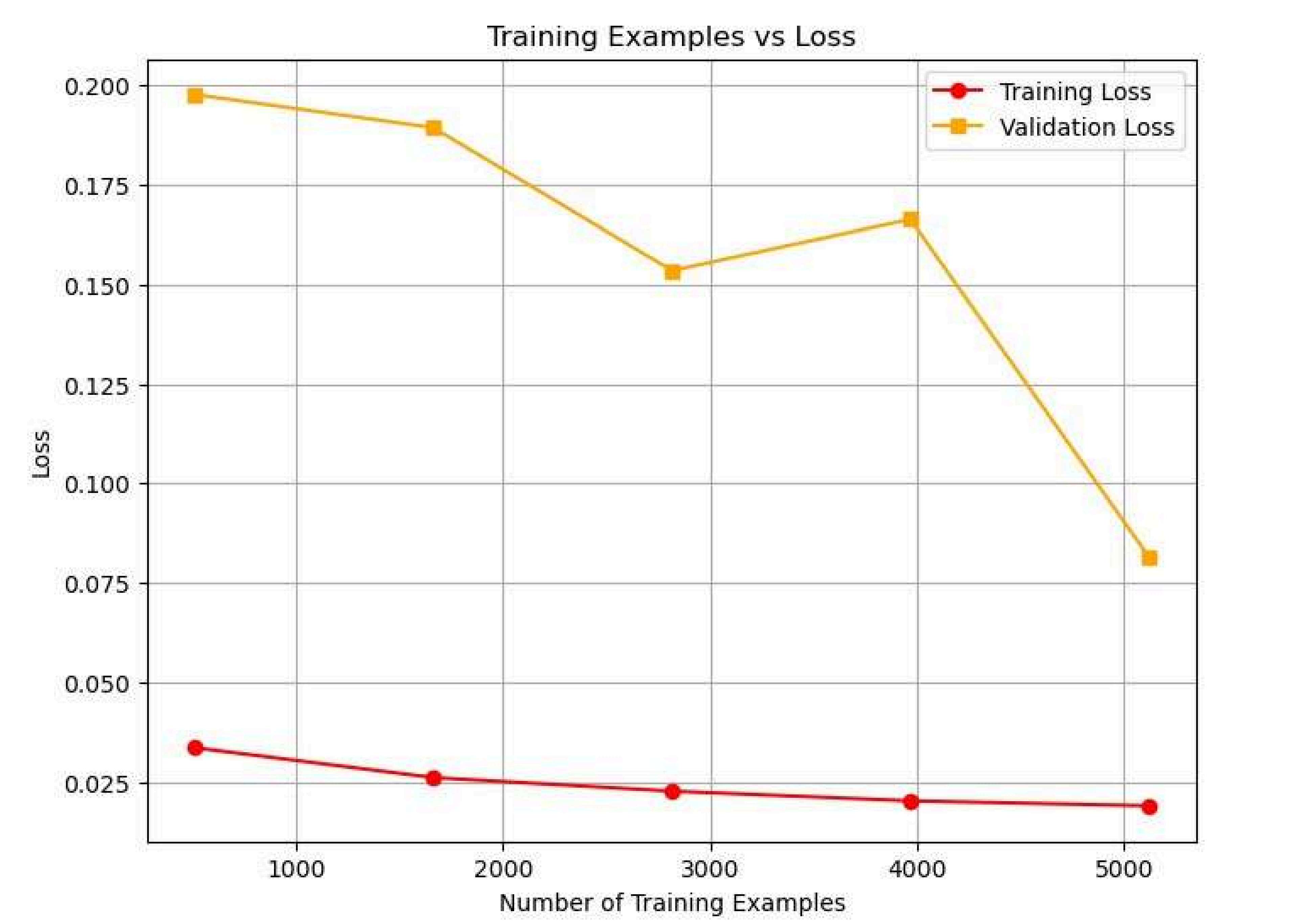
Test Accuracy = 97.5 %

RESULT

Accuracy vs No. of Training Examples



Loss vs No. of Training Examples



CONCLUSION

IMPACT

Implementing this intelligent system will significantly reduce risks, improve energy distribution, and elevate safety standards within smart city infrastructures.

Advanced Integration: Combines IoT, AI, and cloud computing for real-time gas leak detection and prevention.

Enhanced Safety: Automated alerts and remote-control mechanisms rapidly mitigate hazards.

Predictive Maintenance: AI-driven analytics enable proactive pipeline maintenance, reducing downtime and risks.

Emergency Preparedness: Seamless integration with emergency response teams ensures rapid intervention.

Reliability & Efficiency: Redundant safety features enhance reliability, contributing to safer, more sustainable pipeline operations.

THANK YOU FOR LISTENING!