In [2]:
```
import pandas as pd
import numpy as np
```

DATA LOADING & OVERVIEW

In [5]:
```
df=pd.read_csv('sigachi_explosion_dataset (1).csv')
```

In [6]:
```
# Check the Data is loaded or not
df.head()
```

Out[6]:

	Timestamp	Equipment_ID	Temperature	Pressure	Vibration	Dust_Concentration	H
0	2025-06-30 08:00:00+05:30	SD003	94.408034	7.163935	0.200764	32.435659	40
1	2025-06-30 08:00:01+05:30	SD001	72.752410	2.027063	2.320166	35.346418	63
2	2025-06-30 08:00:02+05:30	SD003	70.260798	3.566962	3.222697	53.252898	35
3	2025-06-30 08:00:03+05:30	SD003	61.819810	5.878196	0.421946	33.459203	54
4	2025-06-30 08:00:04+05:30	SD001	112.659157	5.860779	2.549736	34.331946	54

In [12]:
```
# Check the Shape of the dataset
print("The size of the dataset is with rows and columns :",df.shape)
```

```
The size of the dataset is with rows and columns : (10000, 17)
```

In [20]:
```
# Lets check some basic statistics about the Data
display("The Basic Statistics abt data is",df.describe(include='all').T)
```

```
'The Basic Statistics abt data is'
```

	count	unique	top	freq	mean	std	
Timestamp	10000	10000	2025-06-30 08:00:00+05:30	1	NaN	NaN	I
Equipment_ID	10000	3	SD001	3354	NaN	NaN	I
Temperature	10000.0	NaN	NaN	NaN	100.021555	20.003811	26.067
Pressure	10000.0	NaN	NaN	NaN	5.025238	1.951452	
Vibration	10000.0	NaN	NaN	NaN	1.999914	0.973694	
Dust_Concentration	10000.0	NaN	NaN	NaN	30.257085	9.852404	
Humidity	10000.0	NaN	NaN	NaN	49.916228	10.002387	
Ventilation_Flow	10000.0	NaN	NaN	NaN	1.996159	0.500326	0.11
Maintenance_Last_Date	10000	90	2025-04-22	141	NaN	NaN	I
Maintenance_Type	10000	3	routine	3981	NaN	NaN	I
Worker_PPE_Compliance	10000.0	NaN	NaN	NaN	0.7993	0.400544	
Worker_Training_Status	10000.0	NaN	NaN	NaN	0.8507	0.356402	
Operational_Load	10000.0	NaN	NaN	NaN	69.611932	14.708863	14.342
Alarm_Status	10000.0	NaN	NaN	NaN	0.0986	0.298139	
Incident_History	10000.0	NaN	NaN	NaN	0.1392	0.698765	
Explosion_Risk_Score	10000.0	NaN	NaN	NaN	0.153187	0.09128	0.000
Intervention_Priority	10000	3	low	9983	NaN	NaN	I

In [15]:
```
df.columns
```

Out[15]:
```
Index(['Timestamp', 'Equipment_ID', 'Temperature', 'Pressure', 'Vibration',
       'Dust_Concentration', 'Humidity', 'Ventilation_Flow',
       'Maintenance_Last_Date', 'Maintenance_Type', 'Worker_PPE_Compliance',
       'Worker_Training_Status', 'Operational_Load', 'Alarm_Status',
       'Incident_History', 'Explosion_Risk_Score', 'Intervention_Priority'],
      dtype='object')
```

In [13]:
```
print(df['Intervention_Priority'].value_counts(normalize=True))
print("\nUnique Values in Categorical Columns:")
for col in ['Equipment_ID', 'Maintenance_Type', 'Worker_PPE_Compliance', 'Worker_Tr
    print(f"{col}: {df[col].nunique()} unique values")
```

```
Intervention_Priority
low       0.9983
high      0.0011
medium    0.0006
Name: proportion, dtype: float64

Unique Values in Categorical Columns:
Equipment_ID: 3 unique values
Maintenance_Type: 3 unique values
Worker_PPE_Compliance: 2 unique values
Worker_Training_Status: 2 unique values
Alarm_Status: 2 unique values
```

DATA PREPROCESSING

In [21]:
```
# Lets start with Data Cleaning
df['Maintenance_Last_Date'] = pd.to_datetime(df['Maintenance_Last_Date'])
df['Days_Since_Maintenance'] = (pd.to_datetime('2025-06-30') - df['Maintenance_Last
df = df.drop(columns=['Timestamp', 'Maintenance_Last_Date'])
```

In [23]:
```
display(df.head())
```

	Equipment_ID	Temperature	Pressure	Vibration	Dust_Concentration	Humidity	Ventilati
0	SD003	94.408034	7.163935	0.200764	32.435659	40.437999	3
1	SD001	72.752410	2.027063	2.320166	35.346418	63.616152	1
2	SD003	70.260798	3.566962	3.222697	53.252898	35.715801	3
3	SD003	61.819810	5.878196	0.421946	33.459203	54.993909	2
4	SD001	112.659157	5.860779	2.549736	34.331946	54.822119	1

In [24]:
```
 ##  Null Value Handling
null_percentages = df.isnull().mean() * 100
print("\nNull Percentages:")
print(null_percentages)
```

```
Null Percentages:
Equipment_ID              0.0
Temperature               0.0
Pressure                  0.0
Vibration                 0.0
Dust_Concentration        0.0
Humidity                  0.0
Ventilation_Flow          0.0
Maintenance_Type          0.0
Worker_PPE_Compliance     0.0
Worker_Training_Status    0.0
Operational_Load          0.0
Alarm_Status              0.0
Incident_History          0.0
Explosion_Risk_Score      0.0
Intervention_Priority     0.0
Days_Since_Maintenance    0.0
dtype: float64
```

In [25]:
```
for col in df.columns:
    if null_percentages[col] < 15:
        df = df.dropna(subset=[col])
    elif 20 <= null_percentages[col] <= 70:
        if df[col].dtype in ['float64', 'int64']:
            df[col].fillna(df[col].mean(), inplace=True)
        else:
            df[col].fillna(df[col].mode()[0], inplace=True)
    elif null_percentages[col] > 75:
        df = df.drop(columns=[col])
print("\nShape after Null Handling:", df.shape)
```

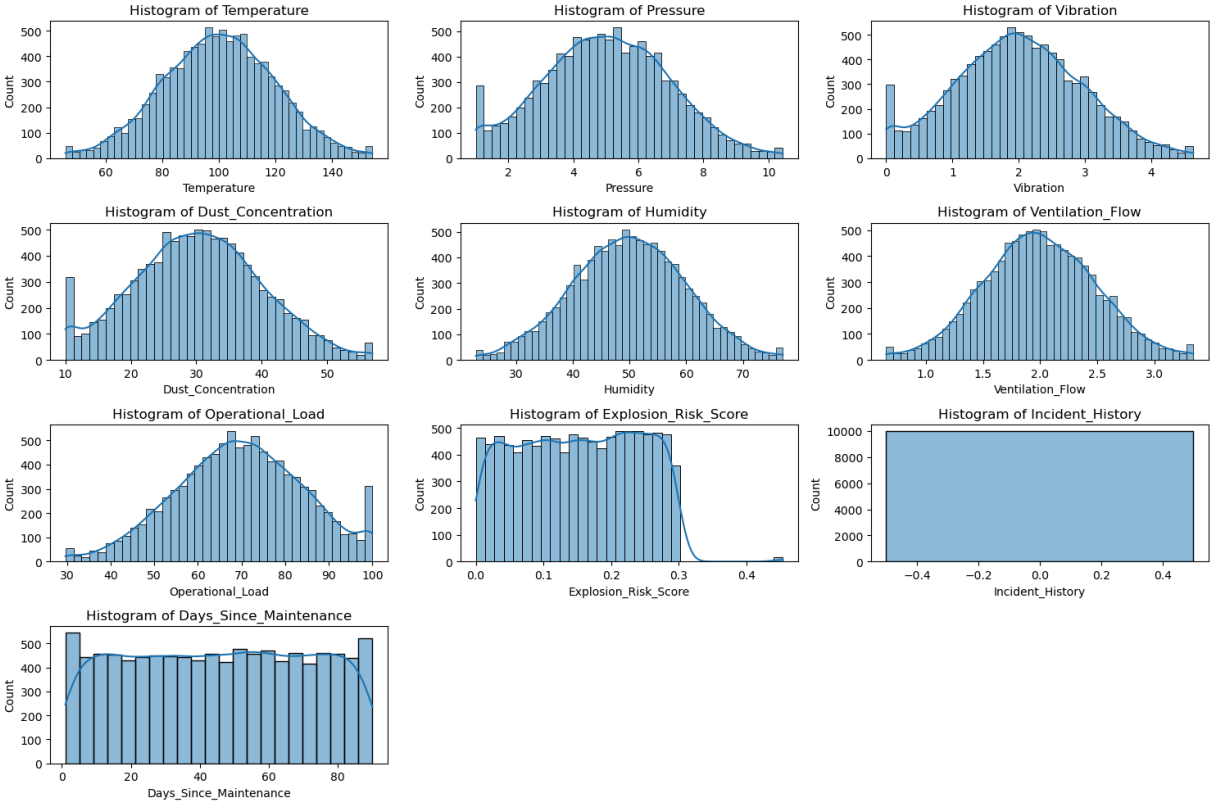
```
Shape after Null Handling: (10000, 16)
```

In [26]:
```
## Outlier Treatment
numerical_cols = ['Temperature', 'Pressure', 'Vibration', 'Dust_Concentration', 'Hu
                  'Ventilation_Flow', 'Operational_Load', 'Explosion_Risk_Score', 'I
                  'Days_Since_Maintenance']
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mean_val = df[col].mean()
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
    df[col] = df[col].where((df[col] >= lower_bound) & (df[col] <= upper_bound), me
```

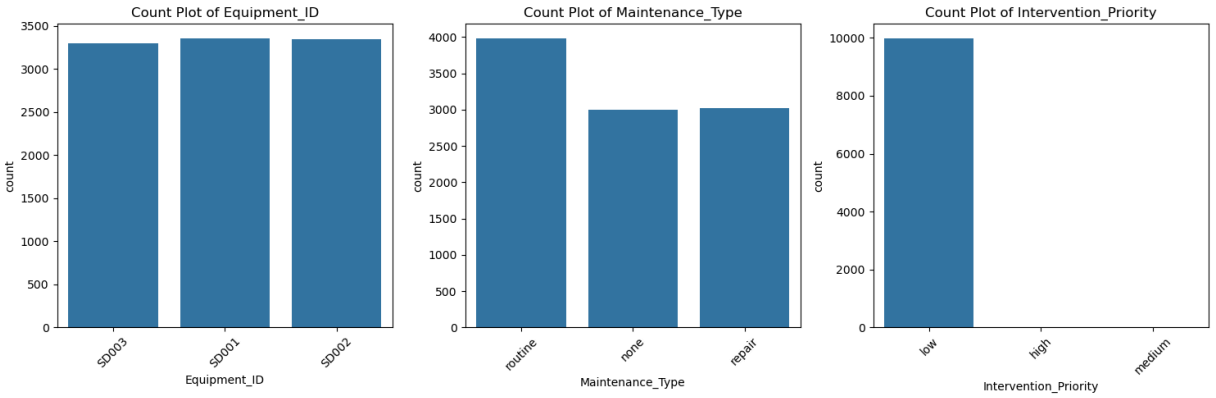
Exploratory Data Analysis (EDA)

In [15]:
```
## Univariate Analysis
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(4, 3, i)
```

```
    sns.histplot(df[col], kde=True)
    plt.title(f'Histogram of {col}')
plt.tight_layout()
plt.show()
plt.close()
```

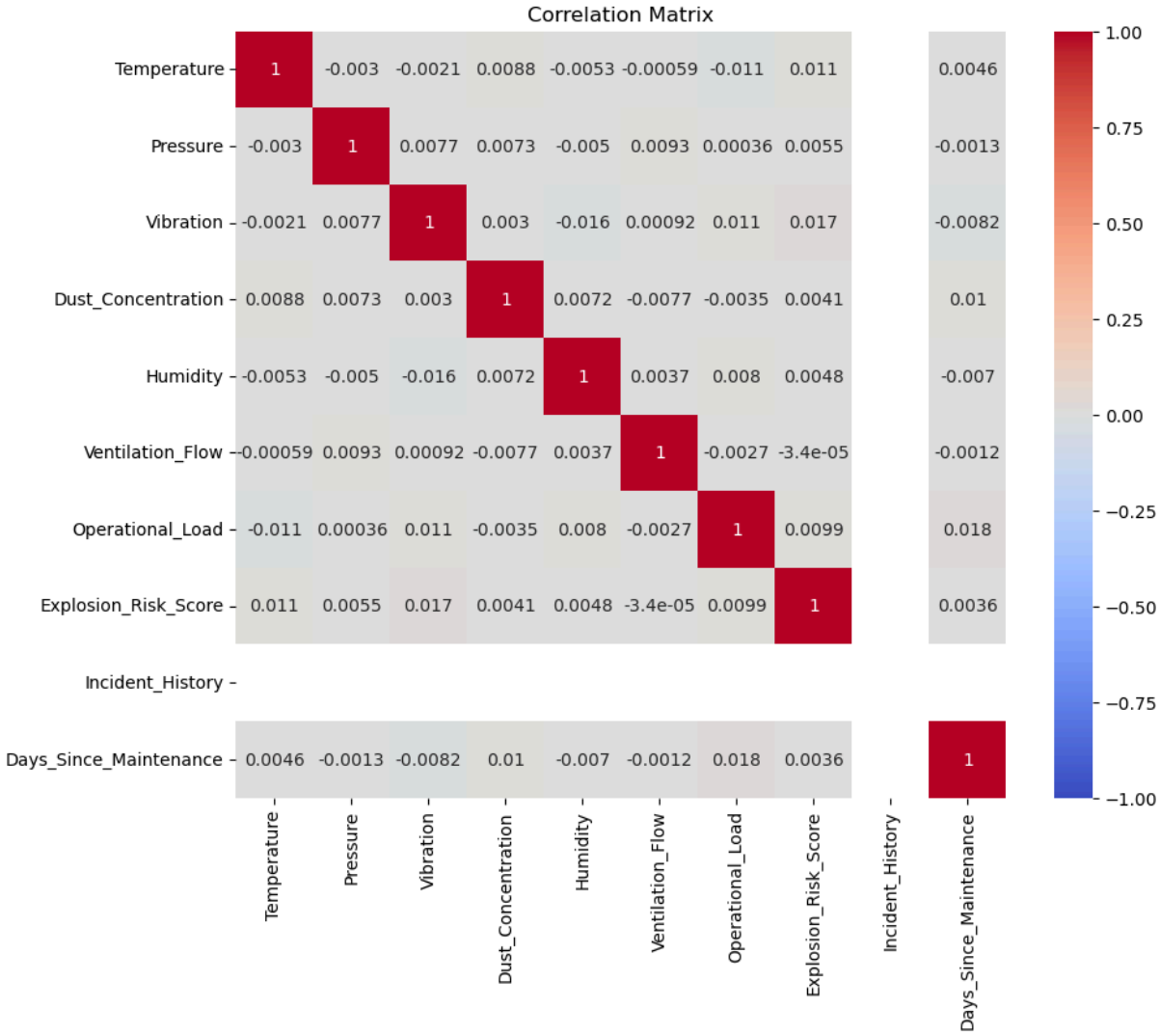


In [16]:
```
plt.figure(figsize=(15, 5))
for i, col in enumerate(['Equipment_ID', 'Maintenance_Type', 'Intervention_Priority
    plt.subplot(1, 3, i)
    sns.countplot(x=col, data=df)
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
plt.close()
```

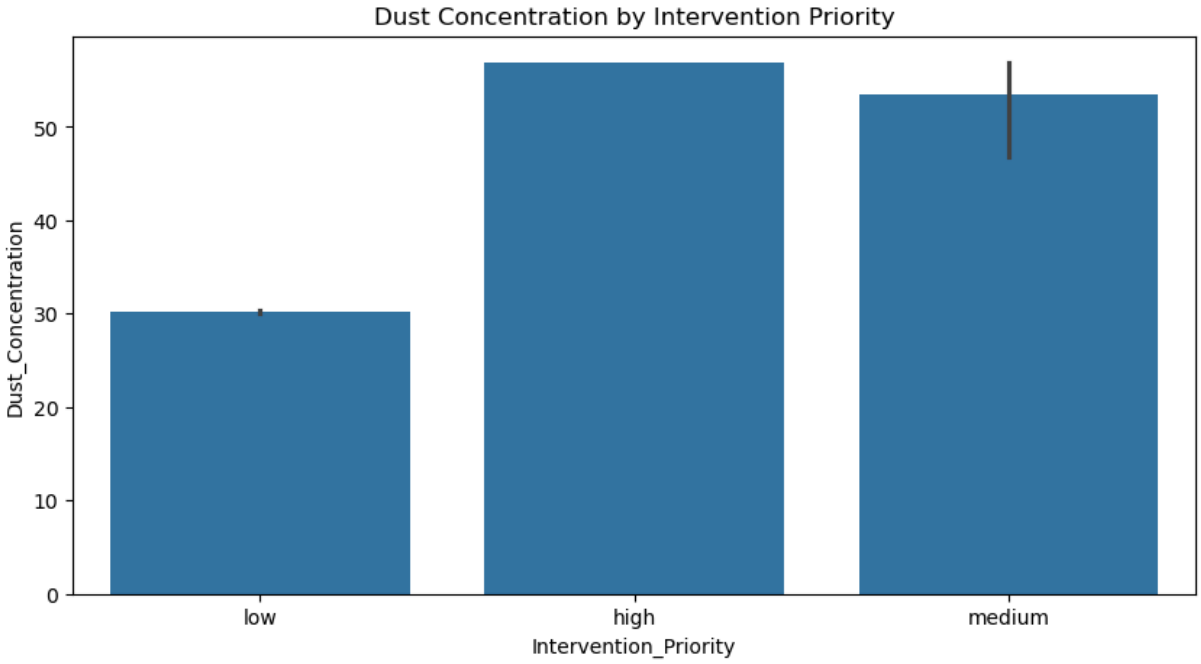


In [19]:
```
## Bivariate Analysis
##Correlation Matrix (Numerical-Numerical)
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(df[numerical_cols].corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1
plt.title('Correlation Matrix')
plt.show()
plt.close()
```



In [20]:
```
# Categorical-Numerical (Bar Plot)
plt.figure(figsize=(10, 5))
sns.barplot(x='Intervention_Priority', y='Dust_Concentration', data=df)
plt.title('Dust Concentration by Intervention Priority')
plt.show()
plt.close()

# Categorical-Categorical (Cross-Tab)
cross_tab = pd.crosstab(df['Equipment_ID'], df['Intervention_Priority'])
print("\nCross-Tab (Equipment_ID vs Intervention_Priority):")
print(cross_tab)
```




```
Cross-Tab (Equipment_ID vs Intervention_Priority):
Intervention_Priority  high   low  medium
Equipment_ID
SD001                     3  3350       1
SD002                     4  3340       3
SD003                     4  3293       2
```

Make Data Ready for ML

In [21]:
```
## Make Data Ready for ML
X = df.drop(columns=['Intervention_Priority'])
y = df['Intervention_Priority'].map({'low': 0, 'medium': 1, 'high': 2})
```

In [22]:
```
# Encoding Categorical Variables
categorical_cols = ['Equipment_ID', 'Maintenance_Type']
# Worker_PPE_Compliance, Worker_Training_Status, Alarm_Status are already binary (0
```

In [26]:
```
# Scaling Numerical Variables
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler,OneHotEncoder
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(drop='first', sparse_output=False), categorical_cols)
    ])
```

In [34]:
```
# Split Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y
print("\nTrain Shape:", X_train.shape, "Test Shape:", X_test.shape)
```

```
Train Shape: (8000, 15) Test Shape: (2000, 15)
```

In [37]:
```
#  Find the Base Model
import xgboost as xgb
from sklearn.pipeline import Pipeline
xgb_model = xgb.XGBClassifier(
    objective='multi:softmax',
    num_class=3,
    random_state=42,
    n_jobs=-1
)
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', xgb_model)
])
```

In [52]:
```
# Baseline (Dummy Classifier)
from sklearn.dummy import DummyClassifier
from sklearn.metrics import f1_score, roc_auc_score
from sklearn.pipeline import Pipeline
dummy = DummyClassifier(strategy='most_frequent', random_state=42)
dummy.fit(X_train, y_train)
y_pred_dummy = dummy.predict(X_test)
print("\nDummy Classifier Macro F1-Score:", f1_score(y_test, y_pred_dummy, average=
```

```
Dummy Classifier Macro F1-Score: 0.3330831456926028
```

In [55]:
```
##Train Base Model
from sklearn.metrics import classification_report
pipeline.fit(X_train, y_train)
y_pred_base = pipeline.predict(X_test)
print("\nBase XGBoost Macro F1-Score:", f1_score(y_test, y_pred_base, average='macr
print("\nBase XGBoost Classification Report:")
print(classification_report(y_test, y_pred_base, target_names=['low', 'medium', 'hi
```

```
Base XGBoost Macro F1-Score: 0.6

Base XGBoost Classification Report:
              precision    recall  f1-score   support

         low       1.00      1.00      1.00      1997
      medium       0.00      0.00      0.00         1
        high       0.67      1.00      0.80         2

    accuracy                           1.00      2000
   macro avg       0.56      0.67      0.60      2000
weighted avg       1.00      1.00      1.00      2000
```

```
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [58]:

```
# Hyperparameter Tuning
from sklearn.model_selection import GridSearchCV, cross_val_score
param_grid = {
    'classifier__max_depth': [3, 6, 9],
    'classifier__learning_rate': [0.01, 0.1, 0.3],
    'classifier__n_estimators': [50, 100, 200]
}
class_weights = {0: 1.0, 1: df['Intervention_Priority'].value_counts()['low'] / df[
                 2: df['Intervention_Priority'].value_counts()['low'] / df['Interve
xgb_model_tuned = xgb.XGBClassifier(
    objective='multi:softmax',
    num_class=3,
    scale_pos_weight=class_weights,
    reg_lambda=1.0,
    reg_alpha=0.5,
    random_state=42,
    n_jobs=-1
)
pipeline_tuned = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', xgb_model_tuned)
])
grid_search = GridSearchCV(pipeline_tuned, param_grid, cv=5, scoring='f1_macro', n_
grid_search.fit(X_train, y_train)

print("\nBest Parameters:", grid_search.best_params_)
y_pred_tuned = grid_search.predict(X_test)
print("\nTuned XGBoost Macro F1-Score:", f1_score(y_test, y_pred_tuned, average='ma
print("\nTuned XGBoost Classification Report:")
print(classification_report(y_test, y_pred_tuned, target_names=['low', 'medium', 'h
```

```
Best Parameters: {'classifier__learning_rate': 0.1, 'classifier__max_depth': 6, 'cla
ssifier__n_estimators': 100}

Tuned XGBoost Macro F1-Score: 0.6

Tuned XGBoost Classification Report:
              precision    recall  f1-score   support

         low       1.00      1.00      1.00      1997
      medium       0.00      0.00      0.00         1
        high       0.67      1.00      0.80         2

    accuracy                           1.00      2000
   macro avg       0.56      0.67      0.60      2000
weighted avg       1.00      1.00      1.00      2000
```

```
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:53:22] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [60]:

```
# Feature Selection with RFE
from sklearn.feature_selection import RFE
rfe = RFE(estimator=xgb_model_tuned, n_features_to_select=10)
pipeline_rfe = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('rfe', rfe),
    ('classifier', xgb_model_tuned.set_params(**grid_search.best_params_))
])
pipeline_rfe.fit(X_train, y_train)
y_pred_rfe = pipeline_rfe.predict(X_test)
print("\nRFE XGBoost Macro F1-Score:", f1_score(y_test, y_pred_rfe, average='macro'
print("\nRFE XGBoost Classification Report:")
print(classification_report(y_test, y_pred_rfe, target_names=['low', 'medium', 'hig
```

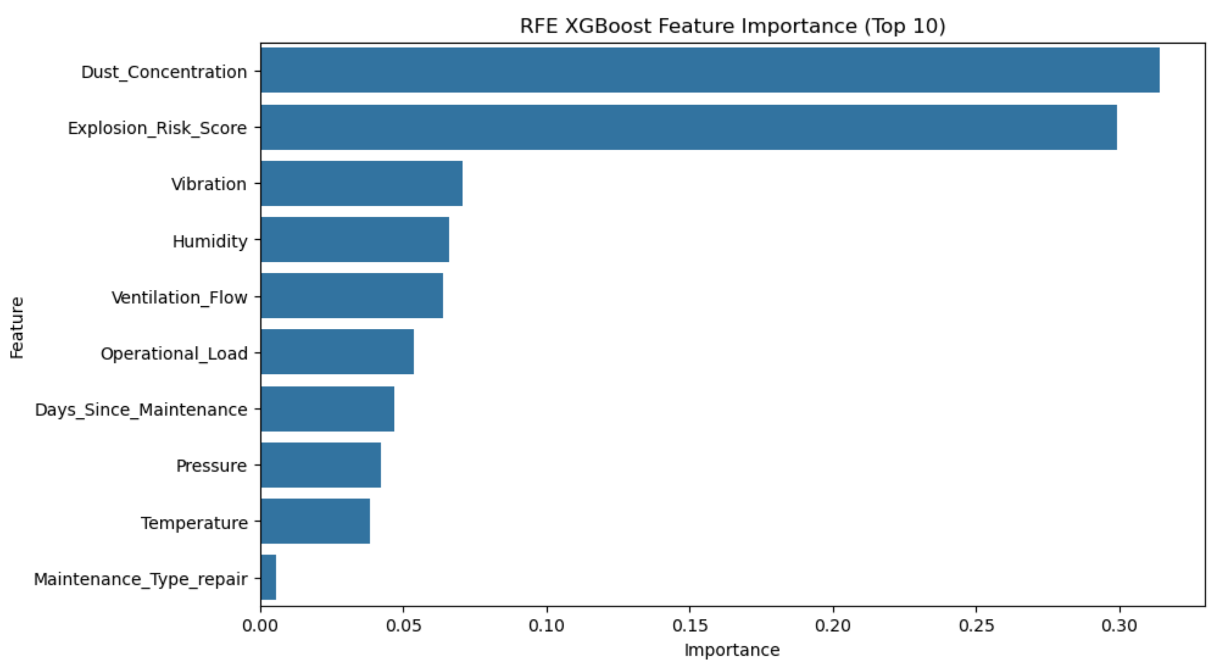
```
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:55:52] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "classifier__learning_rate", "classifier__max_depth", "classifier__n_e
stimators", "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:55:52] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "classifier__learning_rate", "classifier__max_depth", "classifier__n_e
stimators", "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:55:52] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "classifier__learning_rate", "classifier__max_depth", "classifier__n_e
stimators", "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:55:52] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "classifier__learning_rate", "classifier__max_depth", "classifier__n_e
stimators", "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:55:52] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "classifier__learning_rate", "classifier__max_depth", "classifier__n_e
stimators", "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\mppat\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [1
3:55:52] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "classifier__learning_rate", "classifier__max_depth", "classifier__n_e
stimators", "scale_pos_weight" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
RFE XGBoost Macro F1-Score: 0.6

RFE XGBoost Classification Report:
              precision    recall  f1-score   support

         low       1.00      1.00      1.00      1997
      medium       0.00      0.00      0.00         1
        high       0.67      1.00      0.80         2

    accuracy                           1.00      2000
   macro avg       0.56      0.67      0.60      2000
weighted avg       1.00      1.00      1.00      2000
```

```
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\mppat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [62]:

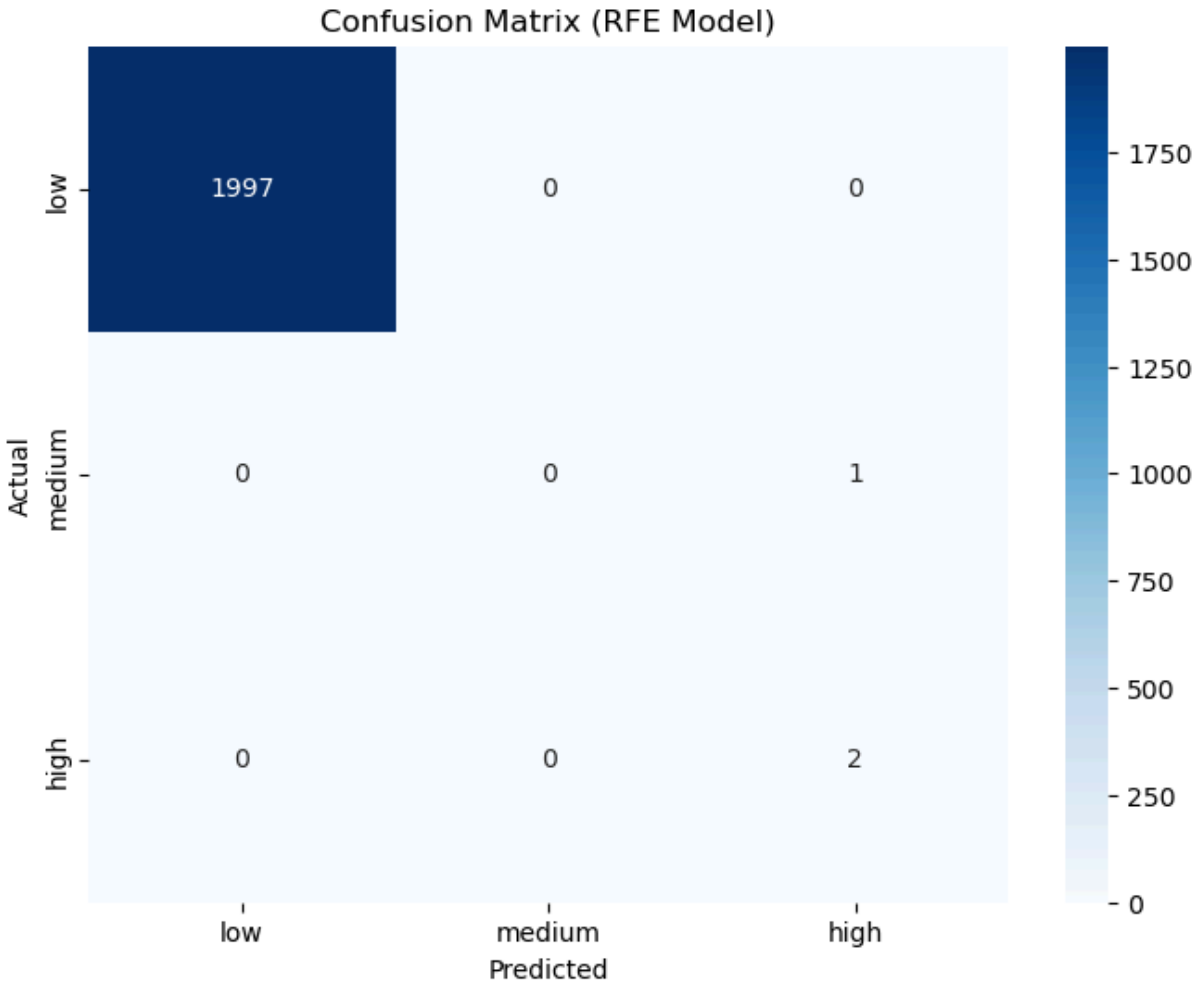
```
# Feature Importance
feature_names = (numerical_cols +
                 pipeline.named_steps['preprocessor']
                 .named_transformers_['cat']
                 .get_feature_names_out(categorical_cols).tolist())
rfe_mask = pipeline_rfe.named_steps['rfe'].support_
selected_features = np.array(feature_names)[rfe_mask]
importances = pipeline_rfe.named_steps['classifier'].feature_importances_
feature_importance_df = pd.DataFrame({
    'Feature': selected_features,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('RFE XGBoost Feature Importance (Top 10)')
plt.show()
plt.close()
```



In [70]:

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_rfe)
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['low', 'medium', 'h
plt.title('Confusion Matrix (RFE Model)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
plt.close()
```



In [ ]: