```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

df = pd.read_csv('Pollutant_Radar.csv')

print("First 5 rows:\n", df.head())

print("\nMissing values:\n", df.isnull().sum())

df.dropna(inplace=True)  # Drop rows with missing values

 df.select_dtypes(include=['object']).columns:

   le = LabelEncoder()

   df[col] = le.fit_transform(df[col])

   label_encoders[col] = le

target_column = 'pollutant_id'

X = df.drop(columns=[target_column])

y = df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train_scaled, y_train)

rf_preds = rf_model.predict(X_test_scaled)

xgb_model = XGBClassifier(n_estimators=100, learning_rate=0.1, use_label_encoder=False,
eval_metric='mlogloss')

xgb_model.fit(X_train_scaled, y_train)

xgb_preds = xgb_model.predict(X_test_scaled)
```

```
print("\nRandom Forest Classification Report:\n", classification_report(y_test, rf_preds))

print("XGBoost Classification Report:\n", classification_report(y_test, xgb_preds))

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

sns.heatmap(confusion_matrix(y_test, rf_preds), annot=True, fmt='d', cmap='Blues')

plt.title("Random Forest Confusion Matrix")

plt.subplot(1, 2, 2)

sns.heatmap(confusion_matrix(y_test, xgb_preds), annot=True, fmt='d', cmap='Greens')

plt.title("XGBoost Confusion Matrix")

plt.tight_layout()

plt.show()

xgb_importance = pd.Series(xgb_model.feature_importances_,
index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(x=xgb_importance, y=xgb_importance.index)

plt.title("XGBoost Feature Importance")

plt.show()
```