```
 ✅ SSTEPTEP 1: Install required libraries
!pip install scikit-learn matplotlib pandas seaborn

# ✅ STEP 2: Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files

# ✅ STEP 3: Upload CSV file
uploaded = files.upload()

# ✅ STEP 4: Load the CSV file
# Replace this with your filename if needed
import io
df = pd.read_csv(io.BytesIO(uploaded[next(iter(uploaded))]))

# ✅ STEP 5: Explore the dataset
print("Dataset Shape:", df.shape)
print(df.head())
print(df.describe())
print(df.info())

# ✅ STEP 6: Preprocessing
# Replace 'target_column' with the actual target variable name (e.g., "AQI")
# Handle missing values
df = df.dropna()

# You might need to change these based on your data
# Make sure 'AQI' is the actual name of the target column in your CSV file
# Check for potential issues like:
# - Case sensitivity: 'aqi' vs 'AQI'
# - Extra spaces: ' AQI' or 'AQI '
# Print the columns to identify the correct target column name:
print(df.columns)

# Replace 'AQI' with the actual target column name from the printed output
target_column = 'AQI'  # Example: Update with the correct column name

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

# Convert categorical variables if any
X = pd.get_dummies(X)
```

```
# ✅ STEP 7: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ✅ STEP 8: Model Training
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# ✅ STEP 9: Prediction
y_pred = model.predict(X_test)

# ✅ STEP 10: Evaluation
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R² Score: {r2:.2f}")

# ✅ STEP 11: Feature Importance
importances = model.feature_importances_
feat_names = X.columns
feat_importance_df = pd.DataFrame({'Feature': feat_names, 'Importance': importances})
feat_importance_df = feat_importance_df.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feat_importance_df.head(10))
plt.title('Top 10 Feature Importances')
plt.show()
```