```
# For Google Colab: Upload file
from google.colab import files
uploaded = files.upload()

# Import the pandas library
import pandas as pd

# Read the uploaded CSV file (update filename as needed)
df = pd.read_csv("AirQualityUCI.csv")  # adjust filename if different
df.head()


# Step 1: Install required libraries (if not already installed)
!pip install pandas numpy matplotlib seaborn scikit-learn tensorflow

# Step 2: Import libraries

import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Step 4: Preprocess the data
df = df.dropna(axis=1, how='all')  # Drop empty columns
df = df.dropna()  # Drop rows with NaNs
df.columns = [col.strip() for col in df.columns]  # Clean column names
df = df.drop(['Date', 'Time'], axis=1)

# Convert to numeric
df = df.apply(pd.to_numeric, errors='coerce')
df = df.dropna()

# Select target column: 'C6H6(GT)'
target_col = 'C6H6(GT)'
data = df[[target_col]].values

# Normalize
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)

# Step 5: Convert to sequences
def create_sequences(data, time_steps=24):
    X, y = [], []
    for i in range(len(data) - time_steps):
        X.append(data[i:i + time_steps])
        y.append(data[i + time_steps])
```

```
    return np.array(X), np.array(y)

TIME_STEPS = 24
X, y = create_sequences(data_scaled, TIME_STEPS)

# Train-test split
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Step 6: Build LSTM model
model = Sequential([
    LSTM(64, input_shape=(TIME_STEPS, 1), return_sequences=False),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

# Step 7: Train model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)

# Step 8: Predict and inverse scale
y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred)
y_test_inv = scaler.inverse_transform(y_test)

# Step 9: Evaluation
print("R2 Score:", r2_score(y_test_inv, y_pred_inv))
print("MSE:", mean_squared_error(y_test_inv, y_pred_inv))

# Step 10: Plot results
plt.figure(figsize=(12,5))
plt.plot(y_test_inv[:200], label="Actual C6H6(GT)")
plt.plot(y_pred_inv[:200], label="Predicted C6H6(GT)")
plt.title("LSTM Air Quality Prediction (Benzene)")
plt.xlabel("Time")
plt.ylabel("C6H6(GT)")
plt.legend()
plt.grid()
plt.show()
```