ASSIGNMENT – 7
Advanced Programming Lab
Name: Utsav Balar
Roll.no. t24cs003

Objective:
Assignment on Anomaly Detection in Air Quality Data. Dataset link:
Objective: Develop an anomaly detection system to identify unusual pollution spikes,
which could indicate hazardous events.
Dataset link: https://archive.ics.uci.edu/ml/datasets/Air+Quality

Anomaly Detection in Air Quality Data

The objective of this project is to develop an anomaly detection system capable of identifying unusual spikes in pollution levels, specifically focusing on carbon monoxide (CO) concentrations. This system aims to detect hazardous events that could pose health risks to individuals and communities. The project utilizes data from the UCI Machine Learning Repository to achieve this goal.

Directions

To begin, ensure that the air quality dataset is downloaded from the UCI repository. After confirming the data source, execute the script to preprocess the data, apply anomaly detection algorithms, and visualize the results. The code performs both statistical analysis and machine learning-based anomaly detection, providing comprehensive insights into air quality variations.

Features

This project focuses on monitoring CO levels as the primary feature for anomaly detection. Key steps involved in the analysis include:

1. **Data Acquisition**: Fetching the air quality dataset from the UCI repository.
2. **Data Preprocessing**:
   Cleaning the data by removing unnecessary columns and handling missing values.
   Creating a time series index using the Date and Time columns for effective time-based analysis.
3. **Exploratory Data Analysis**:
   Visualizing CO levels over time to understand baseline patterns and trends.
   Calculating rolling mean and standard deviation to identify potential anomalies.
4. **Anomaly Detection**:
   Applying a statistical approach using rolling statistics to detect anomalies based on thresholds.
   Implementing the Isolation Forest algorithm, a machine learning technique, to improve anomaly detection accuracy.

Project Structure and Development Process

Environment

The project is designed to run in a Python environment with the following key libraries installed:

1. `pandas` for data manipulation.
2. `numpy` for numerical operations.
3. `matplotlib` for data visualization.
4. `scikit-learn` for implementing machine learning algorithms.
5. `tensorflow` and `keras` can be included for future enhancements or deep learning applications.

Dependencies

Ensure that the necessary Python packages are available in your environment:

1. `pandas` for handling dataframes and preprocessing.
2. `numpy` for numerical computations.
3. `matplotlib` for plotting graphs.
4. `sklearn` for anomaly detection using the Isolation Forest algorithm.

File Structure

1. `readme.md` - Overview and instructions for the project.
2. `src/` - Contains all source code and scripts for data preprocessing and anomaly detection.

Development Process

Throughout the development process, iterative experiments were performed to fine-tune the anomaly detection algorithms. Initial exploratory data analysis provided insights into CO concentration trends, followed by the implementation of both rolling statistics and Isolation Forest methods for identifying anomalies. Each step was carefully documented to ensure reproducibility and clarity.

Results

The anomaly detection system was evaluated using the following metrics:

1. **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)** were calculated to assess the accuracy of the rolling mean method against actual CO levels.
2. The visualizations highlighted CO level trends, with anomalies distinctly marked, providing a clear view of pollution spikes over time.

Results indicated that the Isolation Forest model effectively detected anomalies in CO levels, demonstrating the potential for this approach in real-time air quality monitoring systems.

Future Work

Future enhancements may include:

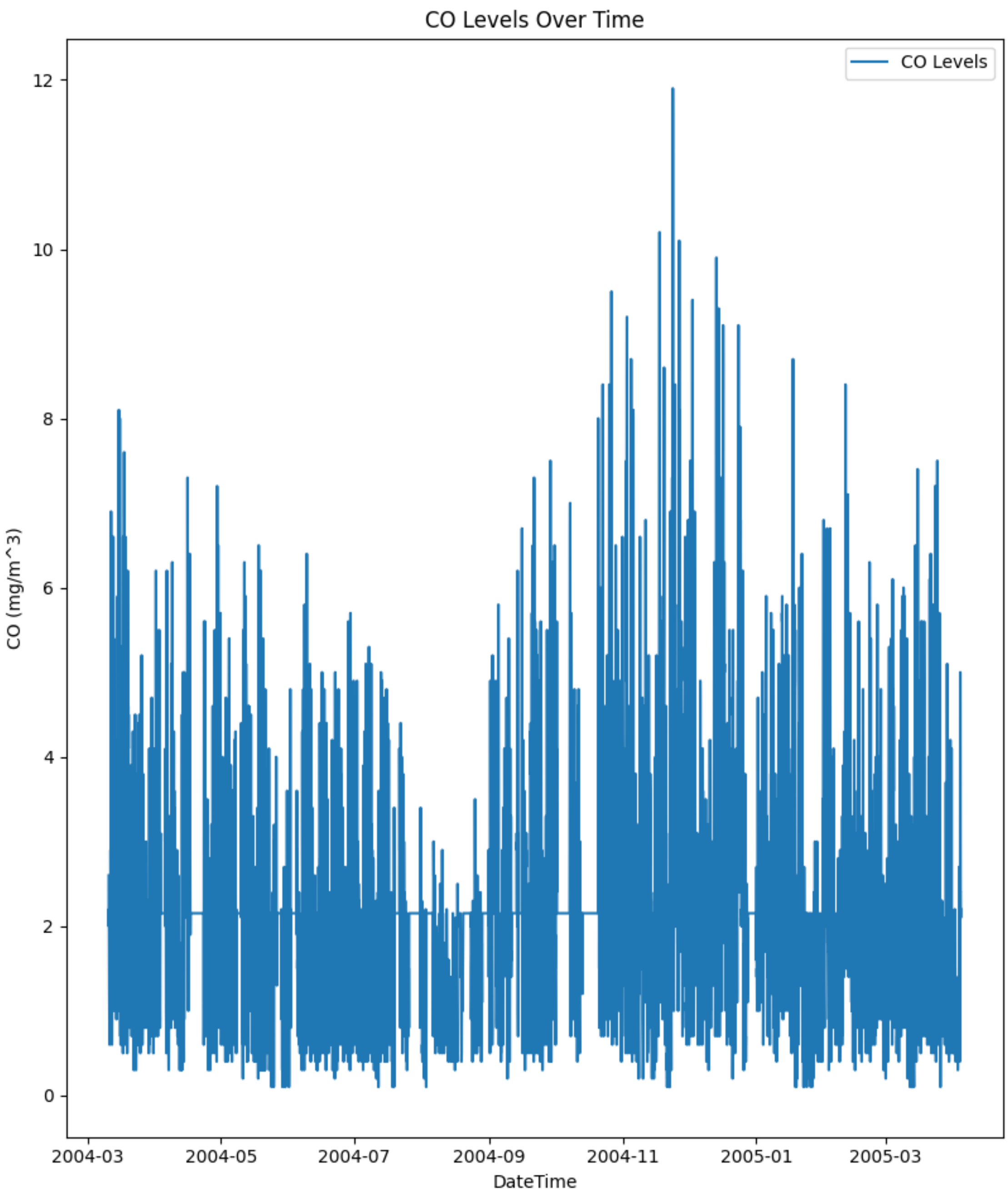
1. Incorporating additional air quality features for a more comprehensive anomaly detection system.
2. Exploring different machine learning algorithms for improved detection accuracy.
3. Implementing a real-time monitoring dashboard that updates with live data, allowing for immediate responses to detected anomalies.

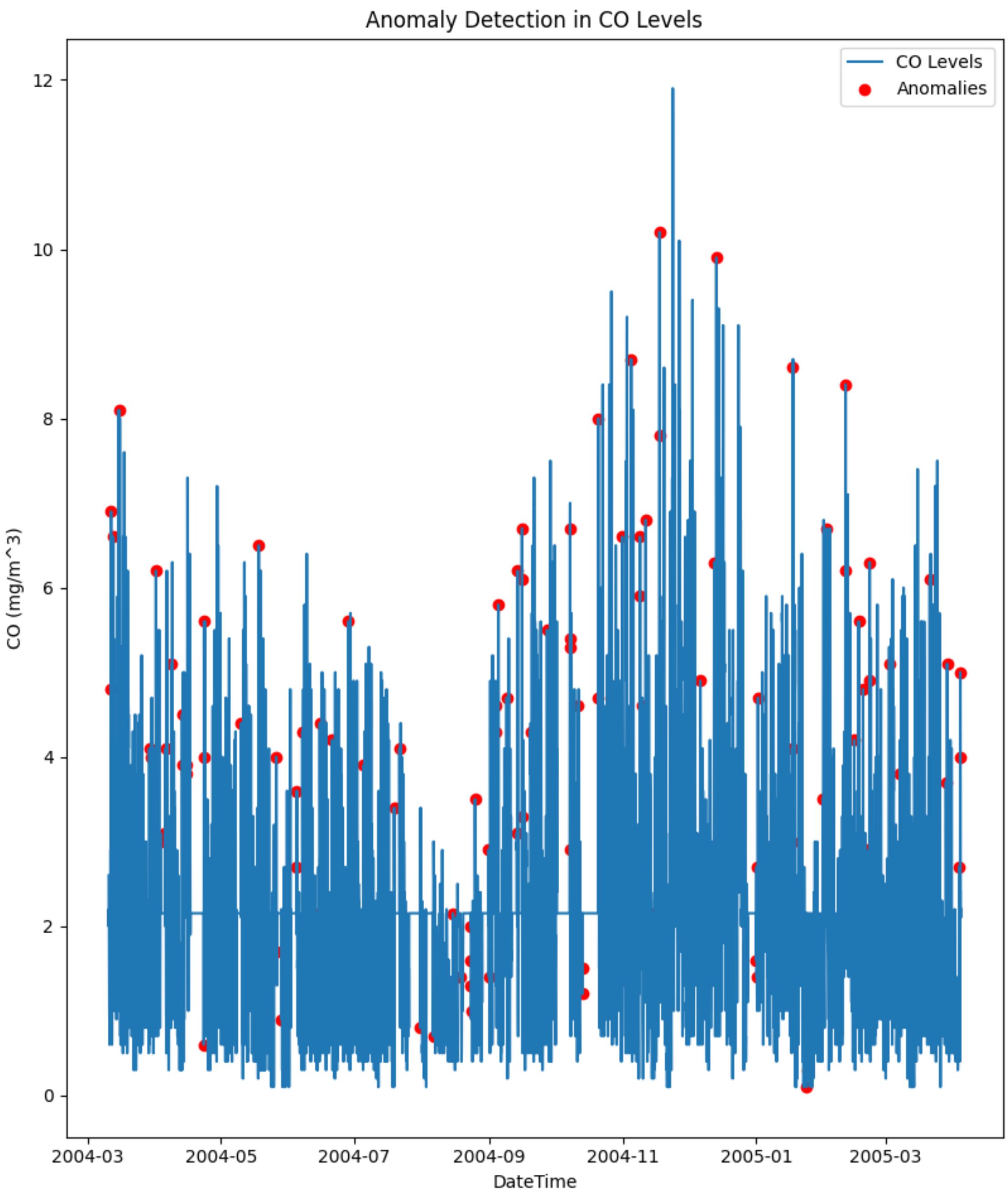
Code:

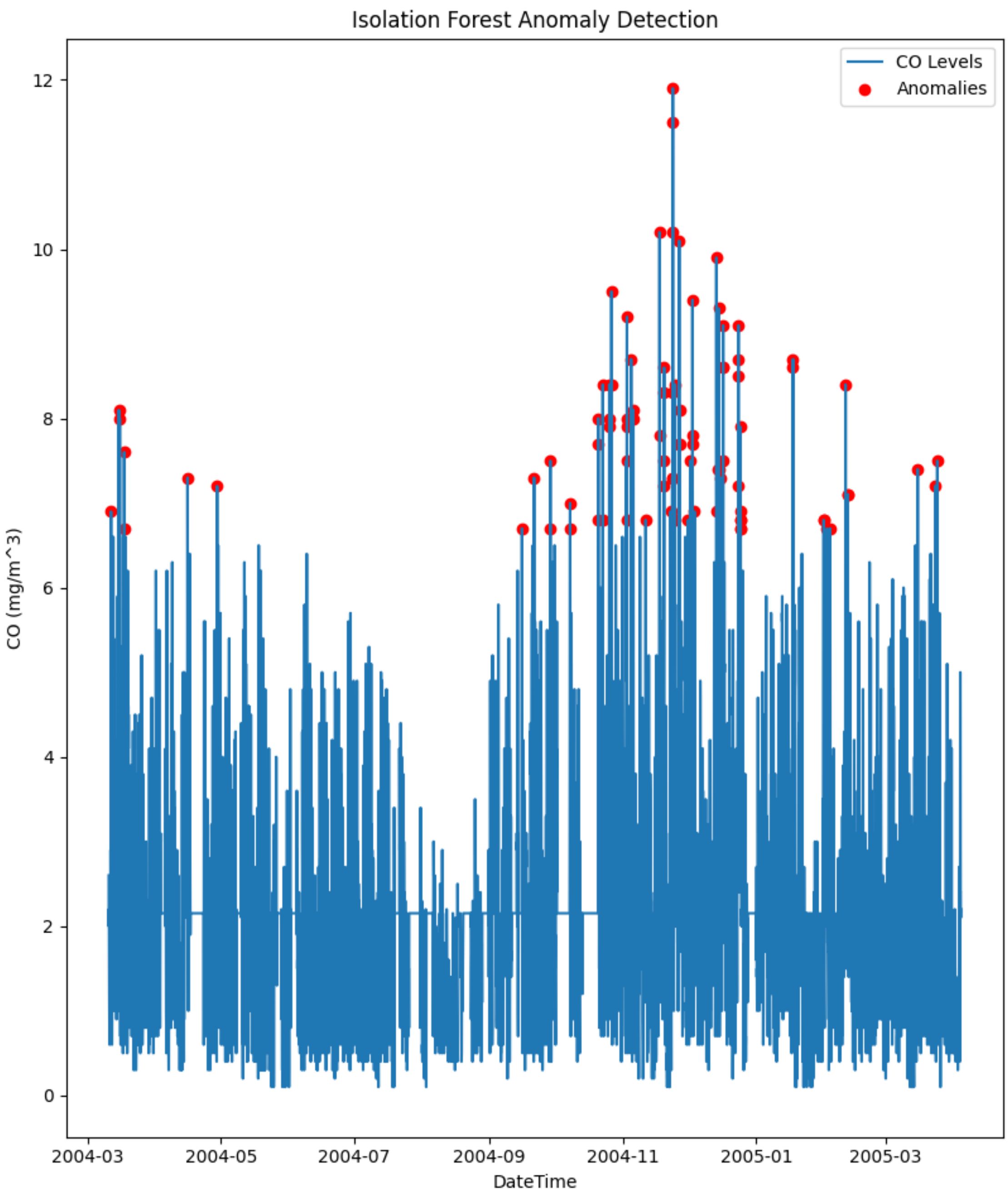
```
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.models import Sequential, load_model
from ucimlrepo import fetch_ucirepo
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error
import warnings

warnings.filterwarnings("ignore")

# Fetch dataset from UCI repository
air_quality = fetch_ucirepo(id=360)

if air_quality.data is None:
    print(air_quality.error)
    exit()

# Combine features and target into a single DataFrame for analysis
df = pd.concat([air_quality.data.features,
air_quality.data.targets], axis=1)

print(df.head())
print(*df.columns)

# ────────── Data Preprocessing ──────────
df.columns = [col.strip() for col in df.columns]  # Remove any
extra whitespace
df["DateTime"] = pd.to_datetime(
    df["Date"] + " " + df["Time"], format="%m/%d/%Y %H:%M:%S"
)
df.set_index("DateTime", inplace=True)
```

```
df["CO(GT)"] = df["CO(GT)"].replace(-200, np.nan)  # Replace
missing value indicator
# Drop unnecessary columns and rows with all NaN values
df = df.drop(columns=["Date", "Time"]).dropna(how="all")
df["CO(GT)"] = df["CO(GT)"].fillna(df["CO(GT)"].mean())


import matplotlib.pyplot as plt

# Plot CO levels over time
plt.figure(figsize=(14, 7))
plt.plot(df.index, df["CO(GT)"], color="tab:blue", label="CO
Levels")
plt.title("CO Levels Over Time")
plt.xlabel("DateTime")
plt.ylabel("CO (mg/m^3)")
plt.legend()
plt.show()


# Calculate rolling mean and standard deviation
rolling_mean = df["CO(GT)"].rolling(window=24).mean()
rolling_std = df["CO(GT)"].rolling(window=24).std()

# Define an anomaly threshold (e.g., 3 standard deviations)
threshold = 3 * rolling_std
anomalies = (df["CO(GT)"] - rolling_mean).abs() > threshold

# Plot anomalies
plt.figure(figsize=(14, 7))
plt.plot(df.index, df["CO(GT)"], color="tab:blue", label="CO
Levels")
plt.scatter(
    df.index[anomalies], df["CO(GT)"][anomalies], color="red",
label="Anomalies"
)
plt.title("Anomaly Detection in CO Levels")
plt.xlabel("DateTime")
plt.ylabel("CO (mg/m^3)")
plt.legend()
plt.show()


from sklearn.ensemble import IsolationForest

# Train an Isolation Forest model
```

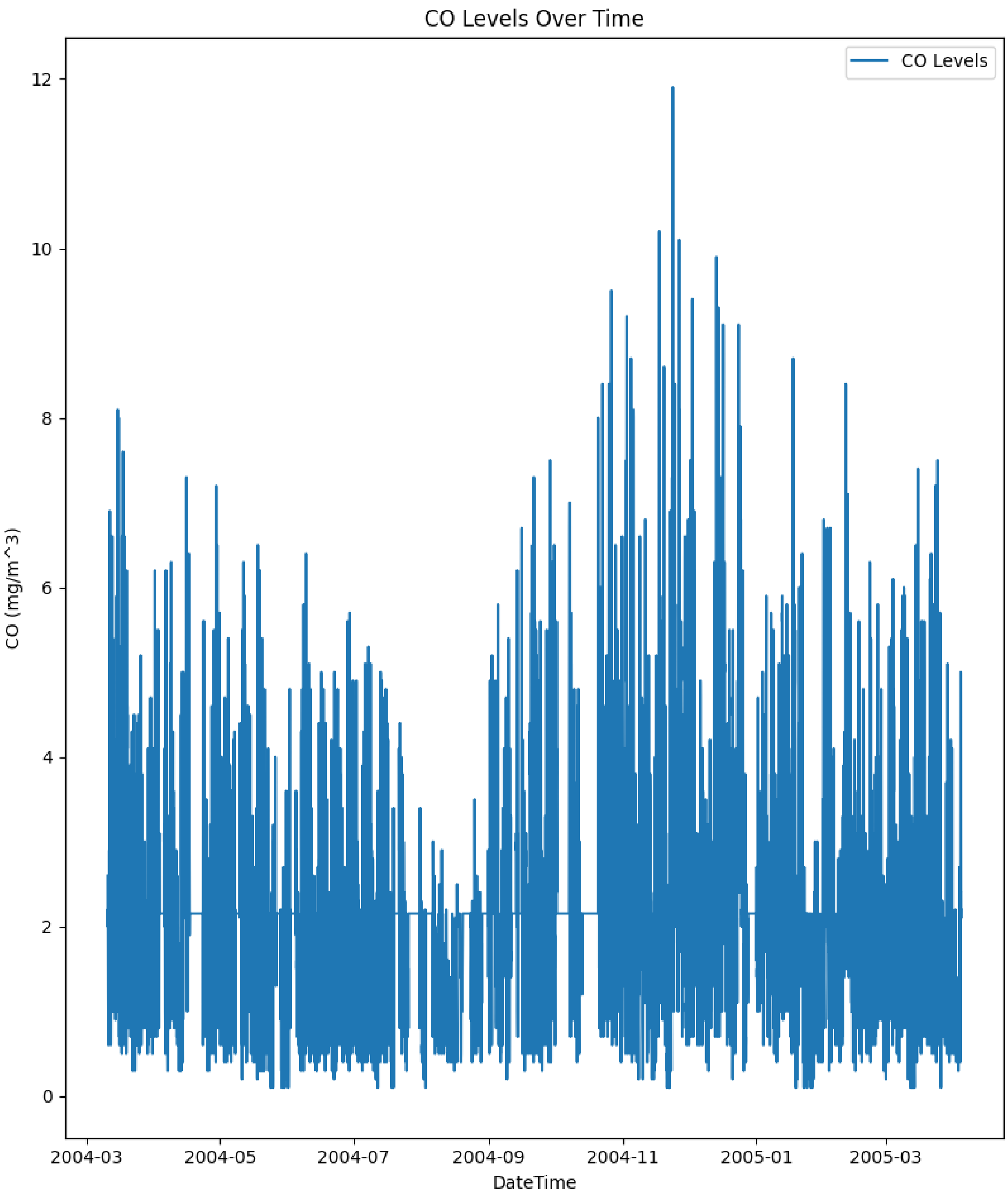
```
model = IsolationForest(contamination=0.01)
df["Anomaly"] = model.fit_predict(df[["CO(GT)"]])


# Plot detected anomalies
anomalies = df[df["Anomaly"] == -1]
plt.figure(figsize=(14, 7))
plt.plot(df.index, df["CO(GT)"], color="tab:blue", label="CO
Levels")
plt.scatter(anomalies.index, anomalies["CO(GT)"], color="red",
label="Anomalies")
plt.title("Isolation Forest Anomaly Detection")
plt.xlabel("DateTime")
plt.ylabel("CO (mg/m^3)")
plt.legend()
plt.show()


# Drop NaN values from the rolling mean and align both series
valid_indices = ~rolling_mean.isna()  # Mask for non-NaN values
mse = mean_squared_error(df["CO(GT)"][valid_indices],
rolling_mean[valid_indices])
mae = mean_absolute_error(df["CO(GT)"][valid_indices],
rolling_mean[valid_indices])

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```


CO Levels Over Time
CO Levels
CO (mg/m^3)
0
2
4
6
8
10
12
2004-03
2004-05
2004-07
2004-09
2004-11
2005-01
2005-03
DateTime


Anomaly Detection in CO Levels
CO Levels
Anomalies
CO (mg/m^3)
DateTime
0
2
4
6
8
10
12
2004-03
2004-05
2004-07
2004-09
2004-11
2005-01
2005-03


Isolation Forest Anomaly Detection
CO Levels
Anomalies
CO (mg/m^3)
DateTime
0
2
4
6
8
10
12
2004-03
2004-05
2004-07
2004-09
2004-11
2005-01
2005-03

```
        Date      Time  CO(GT)  PT08.S1(CO)  ...  PT08.S5(O3)     T    RH      AH
0  3/10/2004  18:00:00     2.6         1360  ...         1268  13.6  48.9  0.7578
1  3/10/2004  19:00:00     2.0         1292  ...          972  13.3  47.7  0.7255
2  3/10/2004  20:00:00     2.2         1402  ...         1074  11.9  54.0  0.7502
3  3/10/2004  21:00:00     2.2         1376  ...         1203  11.0  60.0  0.7867
4  3/10/2004  22:00:00     1.6         1272  ...         1110  11.2  59.6  0.7888

[5 rows x 15 columns]
Date Time CO(GT) PT08.S1(CO) NMHC(GT) C6H6(GT) PT08.S2(NMHC) NOx(GT) PT08.S3(NOx) NO2(GT) PT08.
S4(NO2) PT08.S5(O3) T RH AH
Mean Squared Error: 1.3623281081640821
Mean Absolute Error: 0.8183362109806483
(chb)
utsav …/NITM-T24CS003/Programming_Lab/co-forecasting  master ?  v3.12.7  03:24
```


CO Levels Over Time
CO Levels
CO (mg/m^3)
0
2
4
6
8
10
12
2004-03
2004-05
2004-07
2004-09
2004-11
2005-01
2005-03
DateTime