



***CODE USED TO CREATE DATASET:***

import pandas as pd

import numpy as np

def generate\_multivariate\_time\_series(num\_rows, num\_features, freq='h'):

# Check for potential overflow and adjust frequency if needed

if num\_rows \* (24 if freq == 'h' else 1) > 2\*\*63: # Approximate max safe integer value

freq = 'D' # Change to daily frequency for example

# Generate random data for each feature

data = np.random.randn(num\_rows, num\_features)

# Create a time index

index = pd.date\_range(start='2023-01-01', periods=num\_rows, freq=freq)

# Create a DataFrame with the generated data and time index

df = pd.DataFrame(data, index=index, columns=[f'feature\_{i}' for i in range(num\_features)])

return df

# Set the desired number of rows and features

# Adjust num\_rows based on your needs and potential overflow

num\_rows = 1000000 # Adjust as needed for 1GB (might need adjustment)

num\_features = 5

# Generate the dataset

df = generate\_multivariate\_time\_series(num\_rows, num\_features)

# Save the dataset to a CSV file

df.to\_csv('D:\multivariate\_time\_series\_data.csv')

***CODE USED FOR ANOMALY DETECTION:***

import pandas as pd

import numpy as np

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('your\_dataset.csv.csv')

features = ['feature\_0', 'feature\_1', 'feature\_2','feature\_3','feature\_4',]

# Prepare the data

X = df[features].values

# Isolation Forest

clf\_iso = IsolationForest(contamination=0.01)

clf\_iso.fit(X)

y\_pred\_iso = clf\_iso.predict(X)

df['anomaly\_iso'] = y\_pred\_iso

# Local Outlier Factor

clf\_lof = LocalOutlierFactor(n\_neighbors=20, contamination=0.01)

y\_pred\_lof = clf\_lof.fit\_predict(X)

df['anomaly\_lof'] = y\_pred\_lof

# Visualize the anomalies (e.g., for 'feature\_1')

plt.figure(figsize=(15, 12))

# Isolation Forest

plt.subplot(2, 1, 1)

plt.plot(df.index, df['feature\_1'], label='Feature 1')

plt.scatter(df[df['anomaly\_iso'] == -1].index, df[df['anomaly\_iso'] == -1]['feature\_1'], color='red', label='Anomalies (Isolation Forest)')

plt.xlabel('Timestamp')

plt.ylabel('Value')

plt.title('Anomaly Detection on Feature 1 (Isolation Forest)')

plt.legend()

# Local Outlier Factor

plt.subplot(2, 1, 2)

plt.plot(df.index, df['feature\_1'], label='Feature 1')

plt.scatter(df[df['anomaly\_lof'] == -1].index, df[df['anomaly\_lof'] == -1]['feature\_1'], color='red', label='Anomalies (Local Outlier Factor)')

plt.xlabel('Timestamp')

plt.ylabel('Value')

plt.title('Anomaly Detection on Feature 1 (Local Outlier Factor)')

plt.legend()

plt.tight\_layout()

plt.show()