**Introduction:**

Anomaly detection, also known as outlier identification, is a machine learning problem that seeks to locate data points that drastically differ from the rest of the data. In order to find abnormalities in a given dataset, we will perform anomaly detection in this project utilizing the Local Outlier Factor (LOF) technique.

**Dataset:**

File 'datatest.csv' contains the dataset used for this project. Data from numerous sensors, including temperature, humidity, light, CO2 levels, humidity ratio, and occupancy status, are gathered in this file.

**Data Loading:**

The data from the "datatest.csv" file will be loaded using a Python script that we shall write. A method called load\_data() in the data\_loader.py file utilises NumPy to read the CSV file and returns the loaded data as a NumPy array.

|  |
| --- |
| import numpy as np  def load\_data(file\_name):  # Import the numpy library as np for convenience  data = np.genfromtxt(file\_name, delimiter=',')  # Use the numpy.genfromtxt function to load the data from 'file\_name'  # The data is assumed to be in a CSV format with values separated by commas (delimiter=',')  # The loaded data is stored in the 'data' variable  return data  # Return the loaded data for further use |

**Anomaly Detection:**

The Local Outlier Factor (LOF) method from scikit-learn will be used to detect anomalies. A function called detect\_anomalies(), which accepts the data as input and outputs the anomalies found, is present in the anomaly\_detector.py file.

|  |
| --- |
| from sklearn.neighbors import LocalOutlierFactor  def detect\_anomalies(data):  # Import the LocalOutlierFactor class from the sklearn.neighbors module  # LocalOutlierFactor is used for anomaly detection based on the local density of data points  lof\_model = LocalOutlierFactor(novelty=True)  # Create an instance of the LocalOutlierFactor model with novelty=True  # novelty=True indicates that we are performing outlier detection on new unseen data  # The model will be trained on the given 'data' and later used to detect anomalies in the same data  lof\_model.fit(data)  # Fit the LocalOutlierFactor model to the 'data'  # The model will learn the pattern of normal data points during the training phase  outliers = lof\_model.predict(data)  # Predict the labels for the 'data' using the trained model  # The predicted labels are 1 for inliers (normal data points) and -1 for outliers (anomalies)  anomaly\_indices = np.where(outliers == -1)[0]  # Find the indices of data points that are labeled as outliers (-1) in the 'outliers' array  anomalies = data[anomaly\_indices]  # Extract the data points that are labeled as anomalies based on their indices  return anomalies  # Return the detected anomalies for further analysis or processing |

**Main Script Execution:**

We will run the complete procedure by invoking the methods from data\_loader.py and anomaly\_detector.py in the main.py file.

|  |
| --- |
| # Import necessary functions from data\_loader and anomaly\_detector modules  from data\_loader import load\_data  from anomaly\_detector import detect\_anomalies  # Step 1: Load Data  # Call the load\_data function from data\_loader module to load data from 'datatest.csv'  data = load\_data('datatest.csv')  # Step 2: Anomaly Detection  # Call the detect\_anomalies function from anomaly\_detector module to detect anomalies in the loaded data  anomalies = detect\_anomalies(data)  # Step 3: Save Anomalies to a File  # Open 'anomalies.txt' file in write mode ('w')  with open('anomalies.txt', 'w') as file:  # Write a header indicating that the file contains detected anomalies  file.write("# Detected Anomalies:\n")  # Iterate through each detected anomaly  for anomaly in anomalies:  # Join the values in the anomaly tuple into a comma-separated string and write it to the file  file.write(','.join(str(val) for val in anomaly) + '\n')  # Print a message indicating that anomalies have been detected and saved to the file  print("Anomalies detected and saved to 'anomalies.txt'") |

**Result:**

The discovered anomalies will be stored to the 'anomalies.txt' file in CSV format when the main.py script has completed executing. Each entry in the file indicates an abnormality with its related sensor values.

**Conclusion:**

Anomalies in the dataset were successfully found using the Local Outlier Factor (LOF) technique. A significant tool in several industries, including fraud detection, intrusion detection, and quality control, is anomaly detection.

**References:**

1. Local Outlier Factor (LOF) Algorithm Documentation
2. NumPy Documentation
3. scikit-learn Documentation
4. UCI Machine Learning Repository (Dataset Source)