USER MANUAL

A detailed guide on Bits, Please' Data Anomaly Recognition Tool (DART). This document demonstrates how to install and operate the application.

Contents

Introduction

This project will develop a holistic and innovative solution for detecting abnormal data collected from a smart industrial controller. Using statistical analysis and machine learning, this application will deliver reliable and accurate results in a presentable GUI. It will use the onboard sensors of a CrowPi to continuously gather data and create a machine learning training set.

Requirements

Software Requirements

Whilst the application could potentially work with other software and hardware versions, the application has been developed, tested and verified as fully functional in the following versions:

General system requirements:
Raspbian GNU/Linux 11 (bullseye)
Python 3.9.2
Apache/2.4.53 (Raspbian)
PHP 7.4.28 (ensure mysql/mysqli PHP extension modules are installed and enabled)
10.5.15-MariaDB-0+deb11u1 Raspbian 11

Python3 Package Installation (Pip package) requirements:
mysql-connector-python==8.0.29
numpy==1.22.3
adafruit-circuitpython-dht==3.7.1
scikit-learn==1.0.2
matplotlib==3.5.1

**Please note that these packages will require further packages/dependencies, however these will be installed automatically by Pip*.

Hardware Requirements

CrowPi running Raspbian GNU/Linux 11 (bullseye), with CrowPi hardware drivers installed.
Instructions for the general CrowPi setup and driver installation process can be found at the official CrowPi website via the following link:
https://cdn.shopify.com/s/files/1/0306/3359/0917/files/CrowPi_User_Manual.pdf?v=1631173569

Installation

Clone/download the DART app repository found at https://github.com/Bits-Please-UNSW/ZEIT3118-ITP4-2022 to the hosting device.

Web app front-end installation.

First verify that Apache, PHP, and the mysql/mysqli PHP extension services and modules are installed and running.

Then copy the entire html folder and its contents as seen in the cloned DART app repository folder called 'ZEIT3118-ITP4-2022', found under the 'Dart Web App' subdirectory. Copy this folder to the Apache directory '/var/www'. If a html folder already exists in this directory, completely replace/overwrite it with the new html folder from the cloned DART app repository folder.

MariaDB (MySQL) setup and configuration

First verify that MariaDB has installed successfully and is active/running.

Create a new database called: dart_data

Inside this new dart_data database, create a new table called sensor_data with the following columns and attributes:

```
MariaDB [dart_data]> describe sensor_data;
+---------------+----------+------+-----+---------+-------+
| Field         | Type     | Null | Key | Default | Extra |
+---------------+----------+------+-----+---------+-------+
| DATETIMESTAMP | datetime | NO   |     | NULL    |       |
| TEMPERATURE   | int(11)  | NO   |     | NULL    |       |
| HUMIDITY      | int(11)  | NO   |     | NULL    |       |
+---------------+----------+------+-----+---------+-------+
3 rows in set (0.002 sec)
```

Create the service account used by the DART application, with the default username 'dartuser' and password 'password1', and grant this account full access privileges to the dart_data database, and sensor_data table. (The accounts username and password can be changed where required, however ensure that these changes are updated in the relevant DART app code files, including the Python and PHP files.

Running the DART application

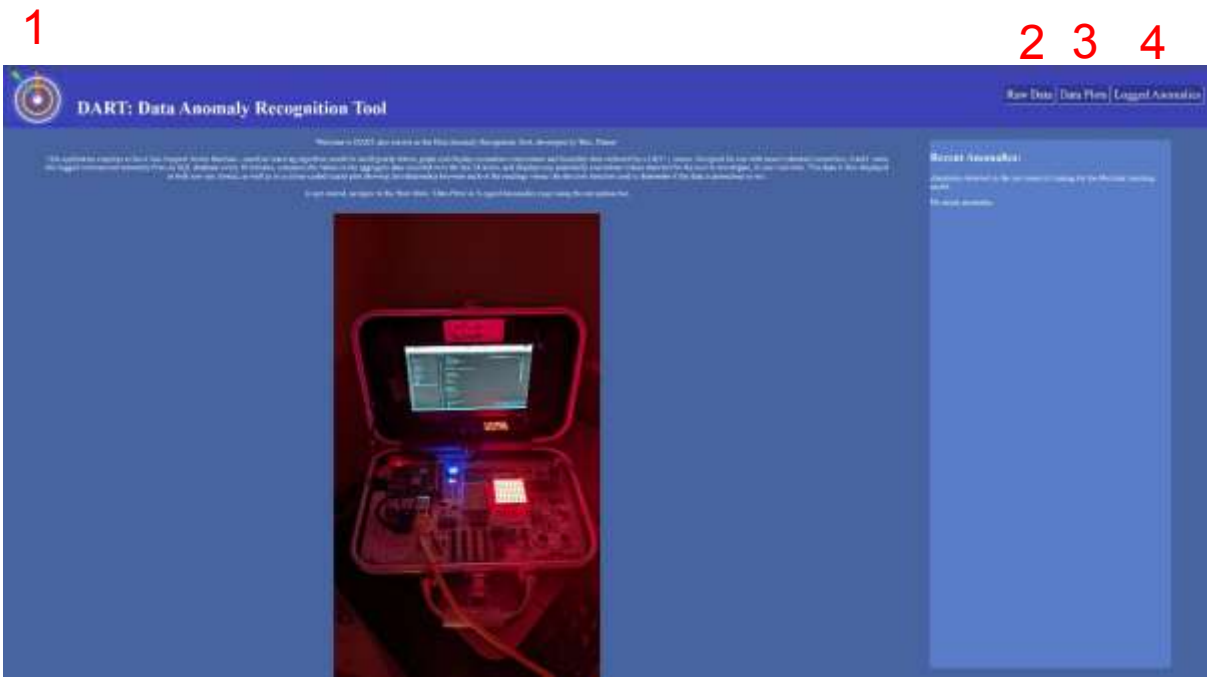
First ensure all previous installation steps and processes are completed, and all required services are installed, configured and running.
To initialise the back-end data engine navigate to the cloned/downloaded DART app repository, and run the 'IntegratedCollectorAndSVM-SQL.py' python script. Note that this can be run in a locally installed IDE, or via the command line via the terminal command:
'Python3 ./IntegratedCollectorAndSVM-SQL.py'

Note that if the analysis thread of this script fails/exits soon after running, allow the script to keep running for several minutes ensuring it is collecting data, indicated by the relevant output to the terminal. Then exit/terminate and restart the script and it should run correctly. Repeat as necessary.

Web App guide and overview

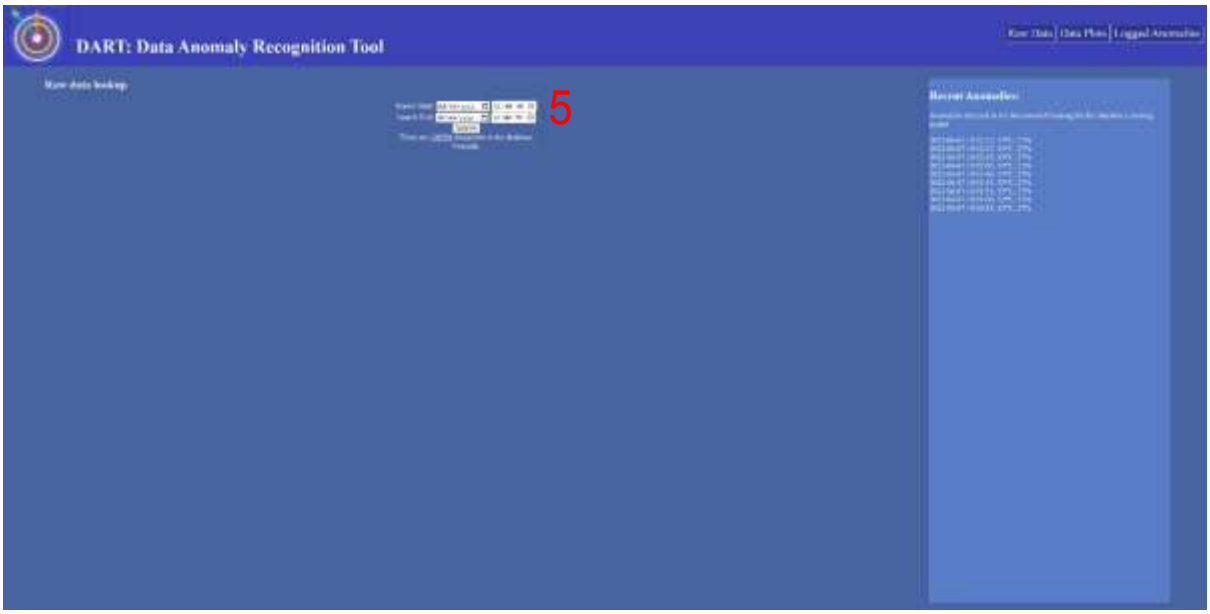
Home Page

The home page provides a quick overview of the application. On each page there is a sidebar which lists recent anomalies from the last training cycle (30 minutes). Clicking on button 1 on any page returns the user to the home page. Clicking on button 2, takes the user to the raw data page. Button 3 takes the user to the data plots page. Button 4 takes the user to the logged anomalies page. The navigation bar is available on every page.

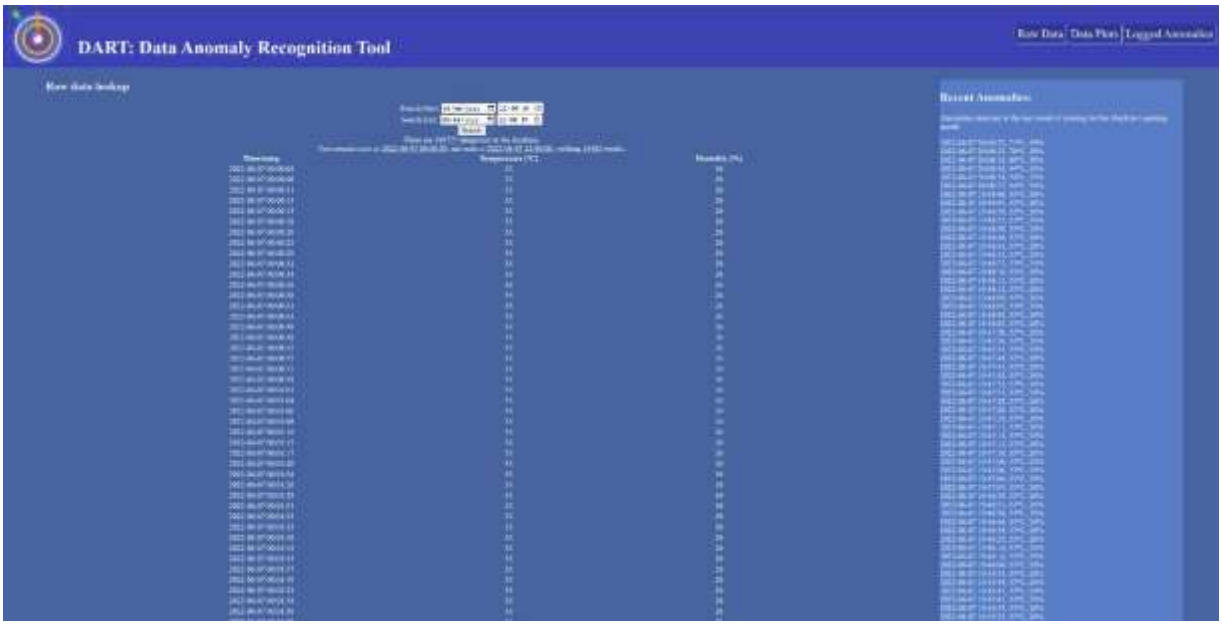


Raw Data Page

The raw data page allows the user to look up any raw data collected within a defined time period. By using the date and time input fields located at position 5, the user can select the desired date range.

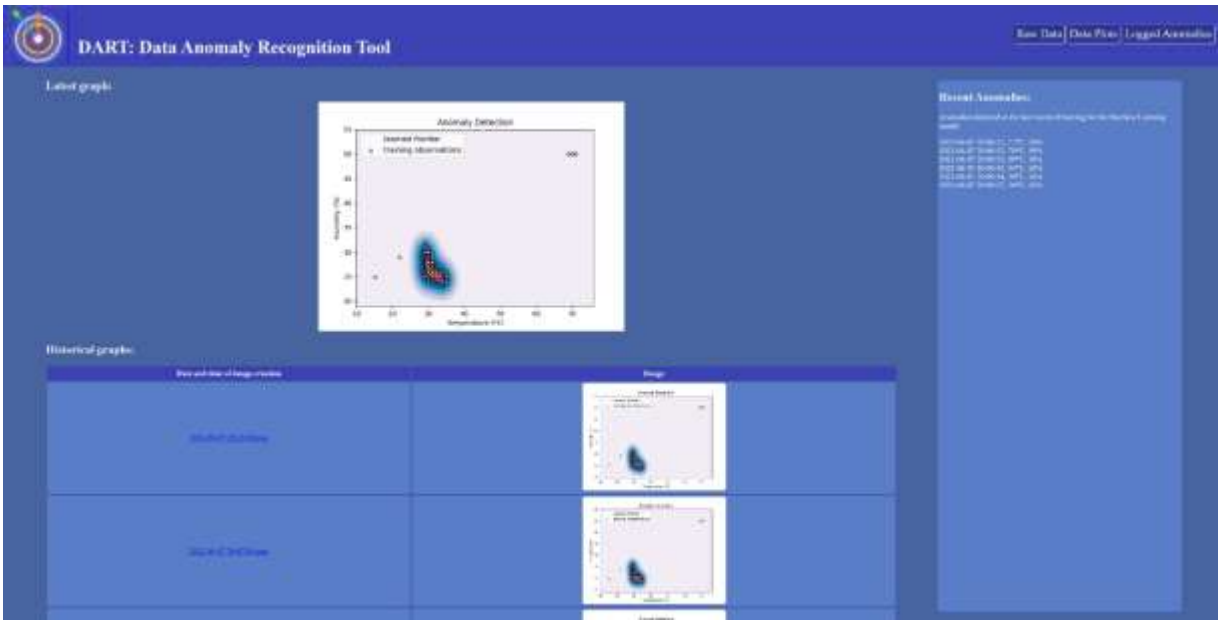


After searching by selecting the ‘Search’ button, the produced output details the full list of relevant raw data, its timestamp, temperature, and humidity.



Data Plots Page

The data plots page displays the graphical output of the collected data. The first centred image is the most recently rendered graph from the last analysis, and below this image is a historical overview of all previously rendered graphs, from most recent to oldest. Clicking on a timestamped hyperlink in the historical graphs section opens a larger image of the graph for display.



Logged Anomalies Page

The logged anomalies page displays all detected abnormal data from the last 24 hours, as determined by the most recent analysis cycle.