DCML-CPS 23-24 course- final project

A Custom Anomaly Detector for Laptops or Wokstations

Aim

The goal of this project is to develop an anomaly detector for a specific system. The system has to be intended as a standalone system (i.e., not a distributed system) for which you have full access to resources, performance monitor from the operating system, and in which you can run custom scripts. Your laptop is fine, tablets or other mobile devices may be fine too albeit they may raise more problems when exercising custom scripts.

Your assignment is to develop a software that runs in your system, silently processes data read at runtime, and triggers an alert whenever an anomaly is detected.

Details

During the course, we have seen and exercised different ways to monitor the performance indicators of a system, and we have analyzed the data points collected to understand if they belong to the normal behavior of a system, or correspond to anomalies. Based on what we did during lab sessions (reusing the provided or developed code is perfectly fine), you should implement a software that:

1. reads specific performance indicators from your system;
2. analyses them through a machine learning algorithm that performs binary classification (normal or anomaly?);
3. writes in a prompt, file, or somewhere else, i) the value of the monitored indicators, and ii) the normal/anomaly prediction of the anomaly detector.

Step 1

This requires to write (or download) a monitor that samples the values of specific performance indicators (e.g., memory usage, active files, open sockets…) of your system at given timeframes (e.g., once a second), and save them in a format that is convenient for analyses. The choice of which indicators to monitor is influenced by the Step 2.

Step 2

Revolves around training an anomaly detector.

Creating Training Data

The anomaly detector needs training data that contains data points (rows, or observations) related to both the normal behavior of your system, or when some anomalies are occurring in the system itself. The dataset has to be labelled: even if you choose an unsupervised anomaly detector, labels are needed to validate “how good” the anomaly detector is. Gathering data related to the normal behavior of the system is easy and just requires to run the monitor from Step 1. To simulate anomalies, you either i) do some strange actions while the system (and the monitor) is running, keeping track of the time in which you do that, or, better ii) craft an injector of errors or attacks that will generate anomalies once activated. The choice of what errors or attacks to simulate is up to the student. Again, this is similar to what we did in many lab sessions and code can be re-used.

Training and Model Selection

Once training data is ready, different ML algorithms should be exercised and compared. The choice of the metric(s) to compare the behavior of different ML algorithms and choose “the best” is up to the student, but has to be well-motivated and sound. Once “the best” algorithm is found, the model has to be stored in a file for Step 3

Step 3

This is the last step in which the model obtained at the end of Step 2 has to be used at runtime to detect anomalies. It requires to load the detector from the file, and feed it with data coming from monitoring activities. Ideally, this step would be very simple and just requires i) loading the detector from file, and then, iteratively ii) read data using the monitor, iii) send data to the detector and log its result.

Submission of the Project

The student has to prepare four different items for submitting the project

- The code, which should be commented wherever the student deems relevant (a ZIP folder). Any programming language is fine.
- The training set used to train anomaly detectors (single CSV file)
- A short video (max 5 minutes) in which the student shows the execution of the detector on his workstation/laptop/system, showing how (and if) it does work
- A PDF document of maximum 5 pages of text (pictures not included, can exceed the 5 pages) that explains:
    - How the monitor is made and why
    - If there was a process to select monitored indicators, and if yes, which one
    - The process of generating the training set, including the injection or simulation of anomalies
    - The process of training, testing and comparing algorithms for anomaly detection, including the motivation behind the choice of the algorithms, the metrics for benchmarking, and parameters tuning (if any)
    - The final integration of the monitor and the detector for creating your runtime anomaly detector.

Criteria for Evaluation

The primary criteria for evaluating submissions is the methodology. The student has to motivate and comment all the relevant choices and has to provide an overall solid work in which each step is detailed in isolation, and how it contributes to reach the overall goal. Students that try different setups of the monitors, features, algorithms, metrics will be positively rewarded and will see their mark increase. However, a project that just takes the code the teacher made available on the GitHub of the course, adding whatever needed at minimum and motivating it well enough is more then acceptable and evaluated positively, but not with the maximum mark.