Anomaly Detection Case

Using Python & Docker

The goal of this case is to implement two anomaly identifiers based on a real-world industrial manufacturing dataset (industrial robot) using scikit-learn and TensorFlow, running in a Docker container.

Part 1: Anomaly Detection (scikit-learn / TensorFlow)

1. Create an empty directory and initialize it with git. Please pay attention to a proper commit log along the case.
2. Read the dataset `20221007_CC_AD_DATA.parquet`.
3. Create the train and "test" split. For the dataset the observations from 11/09/2017 (according to the column `robtime`) are the train and the other ones the test slice.
4. Train 2 Anomaly Detection models on the train split
    a. Isolation Forest (scikit-learn)
    b. Autoencoder (TensorFlow)
5. Use the two trained models and apply them on the "test" split each
6. Check the results of the predictions
    a. plot the predictions
    b. define an appropriate threshold for both model approaches to identify anomalies
    c. make a proposal to implement a completely data-driven calculation of a threshold value
7. Check, if both models identify the same anomalies
8. Use model explanation techniques like SHAP to identify the reasons / features for the anomalies
9. Setup a parameter optimization loop to tune both models by testing different parameterizations and pick the best model afterwards

Part 2: Docker

1. Create a `Dockerfile` that can be used to build a Docker image for Python 3.9, that copies all required project files into the image, including a `requirements.txt` file that is `pip install`-ed during the build of the Docker image.
2. Write a small Shell-Script that contains the `docker run` command to run the container. Mount a volume into the Docker container, so the results of the anomaly detection aren't lost once the container is stopped / deleted.
   There are plus points for following Shell-Scripting best practices (like a short description & usage message if the script is invoked with the `--help` flag, Shebang line, ...)
3. When running the Docker image, a Python entrypoint file is supposed to be executed, which trains the two models, calculates the anomalies of the "test" slice, plots the anomaly curve, and write the plots as images into the mounted volume. Afterwards the Container stops.
4. Setup a second Python entrypoint file, that trains the two anomaly detection models and starts a Flask API afterwards with one endpoint each for the two models. Both endpoints act the same, but with a different model: They accept a dataset (a short time window), calculate the anomaly score(s) for the window and return it.