ICPS Anomaly Detection Project

Samruddhi Khairnar

Introduction :

This project is aimed at building a simulated industrial cyber physical system using the Electra dataset, training an SVM model on it for predicting anomalous points in the dataset, processing and predicting the labels of a simulated stream of incoming data points and showing the output in the form of a dashboard.

Technology Stack :

1. Python3, JupyterLab, Colab.
2. Dash library for Dashboard.
3. Apache Spark for distributed ML model (SVM) training.
4. Apache Hadoop for distributed storage.
5. Apache Kafka for simulated pub-sub producer-consumer model for streaming data points in real time.

References :

1. **Project Architecture** :
   https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8693711

2. **Electra Dataset** :

   https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8926471

Installation :

1. Download the installer (**Copy** : Ctrl+C, **Paste** in WSL : Right Click)

   In WSL, multiple terminals can be used by typing *'ubuntu'* in multiple CMD instances.

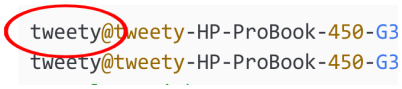
```
tweety@tweety-HP-ProBook-450-G3:~$ sudo apt-get update
tweety@tweety-HP-ProBook-450-G3:~$ sudo apt install python3-pip -y
tweety@tweety-HP-ProBook-450-G3:~$ pip3 install gdown
tweety@tweety-HP-ProBook-450-G3:~$ export PATH=$PATH:/home/<username>/.local/bin
tweety@tweety-HP-ProBook-450-G3:~$ gdown 1WfAGSYFvmjyq4pqwpKkXJCBrP3Pehzqv
```

2. Run the installer *(requires 2 GB data to download all files)* :

```
tweety@tweety-HP-ProBook-450-G3:~$ sh ./icps.sh
```

Intermediate steps (while running the installer) :

i. You'll be asked to enter your ***<username>*** initially.



ii. Later at some point, you will be asked to press *Enter* while the SSH keystore is being generated. Keep pressing *Enter*, to set up passwordless SSH.

iii. After the SSH setup is done by the installer, you'll have to type *'yes'* and press *Enter.*

3. Add the Hadoop bin directory to the PATH, open bashrc :

```
tweety@tweety-HP-ProBook-450-G3:~$ nano .bashrc
```

Type these lines to end of the file (***save*** : ctrl+S -> ***exit*** : ctrl+X) :

```
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
```

4. After the installation is complete, try starting the Hadoop daemons using -

```
tweety@tweety-HP-ProBook-450-G3:~$ start-all.sh
```

Verify whether all Hadoop daemons are running (NameNode, DataNode, Secondary NameNode, NodeManager and Resource Manager), using -

```
tweety@tweety-HP-ProBook-450-G3:~$ jps
```

If all 5 daemons aren't visible, run the following commands and verify that the setup is properly installed. If the issue still persists, run these set of commands : ∞ Electra - EDA.ipynb

```
tweety@tweety-HP-ProBook-450-G3:~$ gdown 1426BQM1k3XBfuvby8dXsbcPhRWw_ZkJb
tweety@tweety-HP-ProBook-450-G3:~$ sh ./hadoop_issues.sh
```

Starting the Background Processes (Hadoop, Spark, Kafka) in one terminal :

```
tweety@tweety-HP-ProBook-450-G3:~$ sh ./start.sh
```

Allow for the startup processing (2-3 mins).

On the first run of the project in second terminal *(run only on the first run, not required later)* :

1. ## Create a directory in HDFS (for aggregating the dataset)

   The Consumer Python script pushes each incoming data point to this file.

   ```
   tweety@tweety-HP-ProBook-450-G3:~$ source .bashrc
   tweety@tweety-HP-ProBook-450-G3:~$ hdfs dfs -mkdir /electra
   tweety@tweety-HP-ProBook-450-G3:~$ hdfs dfs -put ./d.csv /electra/electra_modbus.csv
   ```

2. ## And a Kafka Topic to stream data to and from :

   The Producer Python script pushes a random data point at a fixed interval to this topic and the Consumer reads and pushes it to the HDFS.

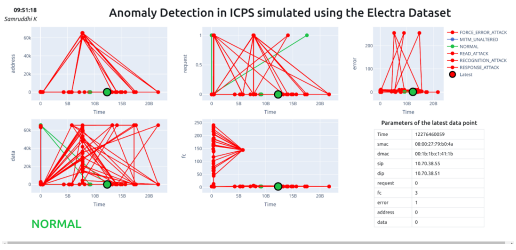
   ```
   tweety@tweety-HP-ProBook-450-G3:~$ $KAFKA_HOME/bin/kafka-topics.sh --create
   --topic electra --bootstrap-server localhost:9092
   ```

Running the Project Scripts (in second terminal) :

```
tweety@tweety-HP-ProBook-450-G3:~$ python3 consumer.py & python3 producer.py &
python3 dashboard.py
```

View the Dashboard using a Browser (allow for 2 mins to startup) :

[127.0.0.1:8050/](127.0.0.1:8050/)



Terminating the Background Processes (allow for 2–3 mins to stop all processes):

Stop Hadoop, Spark, Kafka :

```
tweety@tweety-HP-ProBook-450-G3:~$ sh ./stop.sh
```

Project Overview :

Proposed Architecture for Real-Time ICPS Monitoring System.

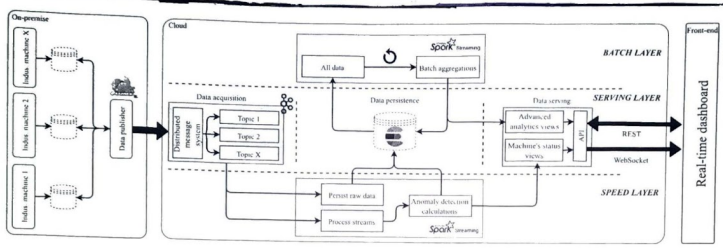


FIGURE 1. Architecture of the proposed ICPS real-time monitoring system.

- Stages :
  1. Data Acquisition
  2. Processing
  3. Persistence
  4. Data Serving.
- Scalable Compute.

- Layers : ① Kafka for Data Acquisition : publish-subscribe model + Flume.
  ② Realtime + Batch processing using Spark Streaming (detect Anomalies).
  ③ Persistence : Distributed storage + querying : Elastic Search + Influx DB.
  ④ Zookeeper : Cloud Resource Management.
  ⑤ Serving : Rest API : Querying + Websocket : Dashboard + communication.

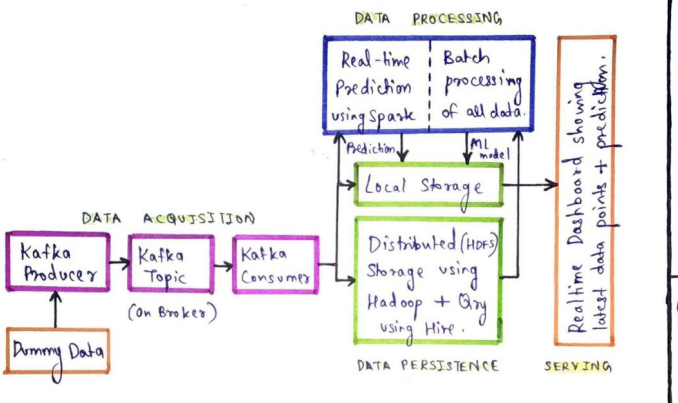
- Detection: ① Single Data Point (SDAD)
  ② Multiple / Batch (MDAD).

- Future: Predict early repair work to increase OEE (Overall Equipment Effectivity)

- Evaluation: (Scale tests) • Input rate • Scheduling delay • Processing time • Delay Time.

- Reference: 10.1109/ACCESS.2019.2911979

Implemented Architecture for ICPS Anomaly Detection System.



- Layers:

① Data acquisition: A stream of data points is simulated using Kafka's producer and consumer.

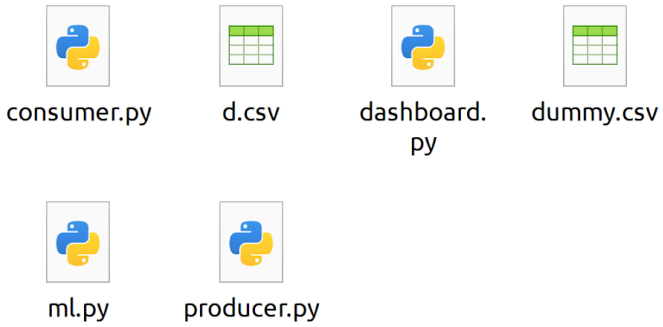
② Persistence: The data points are stored in HDFS, in a distributed manner.

③ Data Processing: The stored data is batch-processed using Apache Spark. Also, the output of it : A trained ML model is saved for predicting the labels of the incoming stream of data points.

④ Data Serving: The latest data points are visualized using a Dashboard.

- Tech Stack: • Hadoop (HDFS, YARN, Hive, Zookeeper) • Spark • Pyspark • Kafka • Python 3 • Plotly-Dash • Colab / Jupyter NB. • Bash • Hosted on AWS EC2 virtual machines.

Folder Structure of the Project : *(The links mentioned contain the details of the source code)*



1. **dummy.csv** : *A chunk of random data points, 400 of each attack label type extracted from the original dataset, to simulate a stream of real time ICPS data.*

   Link to the code snippet showing the sampling :
   Electra - EDA.ipynb

2. **d.csv** :

   This stores the latest 200 data points, to be displayed on the dashboard, the rest are aggregated in the HDFS's `/electra/electra_modbus.csv` for batch training of an SVM model / classifier, using Spark.

3. **ml.py** (For editing, open using : *nano ml.py* command, **Save**: Ctrl+S, **Exit Nano**: Ctrl+X) :

   (Batch training on the electra dataset) This file contains the details of loading the dataset from HDFS's `/electra/electra_modbus.csv` , into a Spark dataframe, preprocessing of the categorical features of the dataset, using OneHotEncoding, transforming the dataset into a PySpark vector, training an SVM model for multi-class classification of the attack labels, using OneVsRest strategy. Then saving the trained model, for the predictions.

   Link to the annotated code, showing the loading and training : Electra - ML.ipynb

4. **producer.py** (For editing, open using : *nano producer.py* command, **Save**: Ctrl+S, **Exit Nano**: Ctrl+X) :

   This script pushes a data point to the '*electra*' topic, created previously after every 10 seconds.

   Link to the annotated code snippet : Electra - Config.ipynb

5. **consumer.py** (For editing, open using : *nano consumer.py* command, **Save**: Ctrl+S, **Exit Nano**: Ctrl+X) :

   This script consumes data from the Kafka topic, obtains a label prediction using the trained SVM model, pushes this data into d.csv (for the dashboard) and the HDFS's `/electra/electra_modbus.csv`, for Batch ML training.

   Link to the annotated code snippet : Electra - Config.ipynb

6. **dashboard.py** (For editing, open using : *nano dashboard.py* command, **Save**: Ctrl+S, **Exit Nano**: Ctrl+X) :

   Displaying 200 data points from the *d.csv*, the latest point read by the consumer, its parameters and predicted label.

   Link to the annotated source code : Electra - Dashboard.ipynb