Spike: Design Data Quality Solution

Description

As a Data Engineering and Data Science team, we need an acceptable way to:

- Track data quality
- Determine if the data thresholds are strict or is a general guideline.
- Graphs to visualize data quality trends.
- Provide alerts on bad data to ensure ML predictions are accurate.

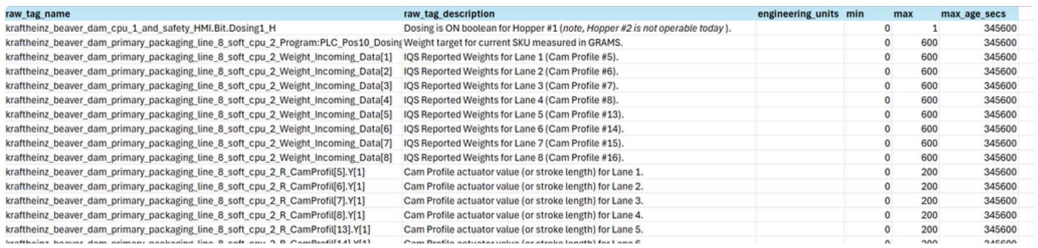
Acceptance Criteria

- Data Quality requirements are captured and solution design is documented in an ADR and approved by Data Science and Data Engineering teams.
- Identify clear metrics to be measured, ranges?, frequency? and criteria for alerting.
- New stories are added to DTGA-88: WS2: Data Quality IN PROGRESS to implement the solution.

Findings

What can we gather from the data in the present state?

Snapshot of data format WS1_ML_RawData_AllTags_Catalog_20240605.xlsx :



raw_tag_name	raw_tag_description	engineering_units	min	max	max_age_secs
kraftheinz_beaver_dam_cpu_1_and_safety_HMI.Bit.Dosing1_H	Dosing is ON boolean for Hopper #1 (*note, Hopper #2 is not operable today*).		0	1	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Program:PLC_Pos10_Dosing	Weight target for current SKU measured in GRAMS.		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[1]	IQS Reported Weights for Lane 1 (Cam Profile #5).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[2]	IQS Reported Weights for Lane 2 (Cam Profile #6).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[3]	IQS Reported Weights for Lane 3 (Cam Profile #7).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[4]	IQS Reported Weights for Lane 4 (Cam Profile #8).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[5]	IQS Reported Weights for Lane 5 (Cam Profile #13).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[6]	IQS Reported Weights for Lane 6 (Cam Profile #14).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[7]	IQS Reported Weights for Lane 7 (Cam Profile #15).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[8]	IQS Reported Weights for Lane 8 (Cam Profile #16).		0	600	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_R_CamProfil[5].Y[1]	Cam Profile actuator value (or stroke length) for Lane 1.		0	200	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_R_CamProfil[6].Y[1]	Cam Profile actuator value (or stroke length) for Lane 2.		0	200	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_R_CamProfil[7].Y[1]	Cam Profile actuator value (or stroke length) for Lane 3.		0	200	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_R_CamProfil[8].Y[1]	Cam Profile actuator value (or stroke length) for Lane 4.		0	200	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_R_CamProfil[13].Y[1]	Cam Profile actuator value (or stroke length) for Lane 5.		0	200	345600
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_R_CamProfil[14].Y[1]	Cam Profile actuator value (or stroke length) for Lane 6.		0	200	345600

Currently there are three main columns we are referencing to determine if the data obtained in accurate:

- **min** - minimum accepted value for that tag.
- **max** - maximum accepted value for that tag.
- **max_age_secs** - maximum allowed time that has passed since last data received.

How do we determine the metrics to track?

Metric #1: Value Range

We can determine data quality by evaluating datapoints based on the min and max column values.

Example:

If we receive a data value of 250 F for the tag: *kraftheinz_beaver_dam_processing_mix_pasteurizer_mix_pasteurizer_Mix_Set_Temperature,* then we can evaluate that data point as faulty data since the min value is 32 F and max value is 212.2 F and this datapoint falls outside that range.

Metric #2: Missing Data

We can keep track of the max_age_secs column and determine if we have received an updated value from the data source for that particular tag. This value needs to fall within the max_age_secs threshold, otherwise we can create a metric to track tags that are missing data beyond the max_age_secs value.

Example:

If we do not receive a value for tag - kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[2]

in 345600 secs (max_age_secs threshold). We can create a metric to track this scenario taking place for this tag.

Metric #3: Data type validation

This metric can be used to validate if the data value received from the data source is of the correct datatype.

Example:

If the data received for tag - kraftheinz_beaver_dam_cpu_1_and_safety_HMI.Bit.Dosing1_H is True or False (Boolean or string) instead of 0 or 1 (integer). We can create a metric to track if the data received is in the correct format.

How do we analyze/visualize the metrics we have gathered?

Tables:

We have the following data: tag name, failure count, and failure reason. From this, we can create one list of tuple with the following schema:

`[(tag_name, failure_reason)]`

From this, we can create tables that help us visualize the errors and even sort and manipulate the data to match our needs.

Example:

Tag Name	Total Failure Count
kraftheinz_beaver_dam_processing_mix_pasteurizer_mix_pasteurizer_Mix_Set_Temperature	3
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[2]	2

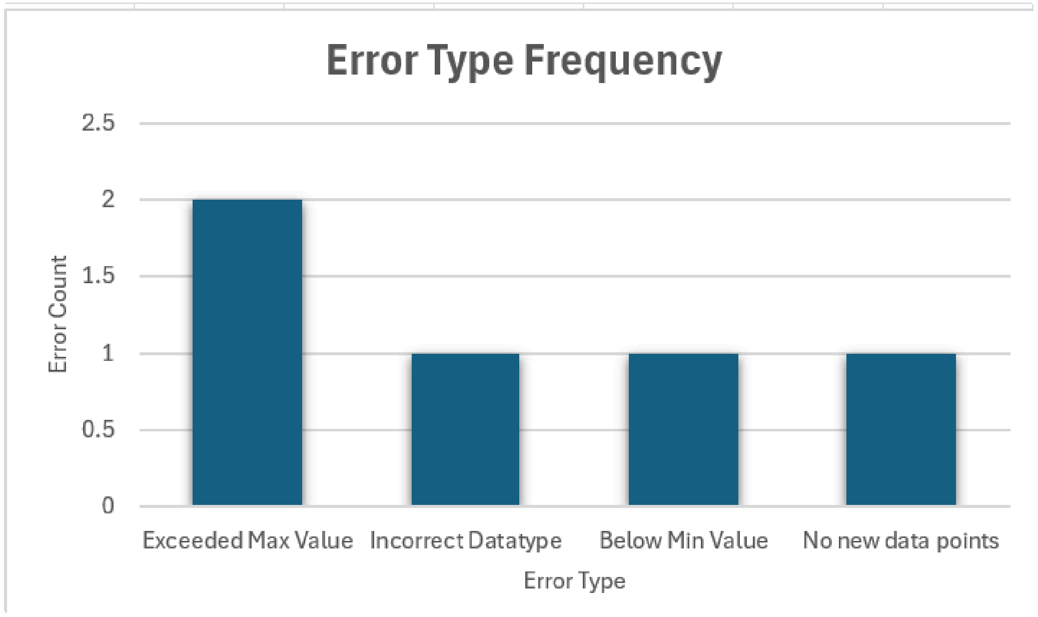
Tag Name	Failure Reason
kraftheinz_beaver_dam_processing_mix_pasteurizer_mix_pasteurizer_Mix_Set_Temperature	Exceeded Max Value
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[2]	Incorrect Datatype
kraftheinz_beaver_dam_processing_mix_pasteurizer_mix_pasteurizer_Mix_Set_Temperature	Below Min Value
kraftheinz_beaver_dam_primary_packaging_line_8_soft_cpu_2_Weight_Incoming_Data[2]	No new data points

kraftheinz_beaver_dam_processing_mix_pasteurizer_mix_pasteurizer_Mix_Set_Temperature	Exceeded Max Value

Graphs:

There are roughly 1170 tags (as per the excel file above). Due to this large amount, we cannot have graphs for each and every tag individually.

We can visualize cases when there are failure instances in any of the three metrics (min-max, time, datatype):



Optional graph: We can also add a dynamic chart that will update with the tags that contain the highest error counts.

Brief notes:

- **Alarming**: We can additionally configure an alarming system to trigger an alarm on specific thresholds we select. The alarming method can vary depending on the user needed: Email, Pager, MS team's channel alert, etc. An example alarm scenario could be: "Send an email to all WS2 engineers when the Exceeded Max Value error type for all tags is greater than 10".
- **Invalid Data:** The current agreed approach on invalid data would be to not modify the incoming data (no removal of bad data) but rather pass the JSON object of results (showing which datapoints are invalid) to the inference function as a parameter. This will decouple the validation check and the action to take since there may be instances where some bad data is acceptable to continue running inference. The validation check would also be conducted prior to pivoting the data.

Which tools should we use for data quality checks?

Great Expectations

Great Expectations is an open-source Python library designed for data quality checks, validation, and documentation. Can be used to create assertions on the data to ensure the data is in the expected format.

Home | Great Expectations

POC for Great Expectations: great-expectations-sample

Grafana (Future)

Grafana is an open-source analytics and monitoring platform designed to help users in visualizing and understanding complex data. It allows you to query, visualize, alert, and explore metrics, logs, and traces.

A Grafana dashboard would be a great tool to monitor the failure rates in a single place. Since this setup will need adjustments on alarming values after the data is validated, we will not be prioritizing this for the next few sprints.

Grafana | Query, visualize, alerting observability platform

Resources:

How to ensure data quality with Great Expectations | by Tomáš Sobotík | Snowflake Builders Blog: Data Engineers, App Developers, AI/ML, & Data Science | Medium

great-expectations/great_expectations: Always know what to expect from your data. (github.com)

Get started with Great Expectations and Databricks | Great Expectations