Project monitoring

Metrics & checks methodology

To monitor the project, we use the metrics & checks Dataiku feature.

For this, we defined two classes in the *dataiku__metrics_checks.py* module of the code base. This means that a check on an element (dataset or folder) can be **called from any location** be it a check python probe, a recipe or a notebook. For better clarity and error tracking, we still prefer to instantiate the classes from the element itself so the scenario can be stopped **as soon as a check returns an error**.

Common initialization

The following *Metric* and *MetricsAndChecks* classes initialization is mostly done through usage of the **common** *metric_init_helper* function.

Both classes are instantiated on a specific element of the project, be it a dataset or a folder. The function handles the discrepancies between folders and datasets to provide attributes in the same way for both.

Note that it currently only works for folders stored in GCS or another external location, as we do not use internal Dataiku storage (in order to not fill the DSS disk space).

Metric class

The class contains functions that are related to a **specific metric**, be it for example retrieving the last value or n last values of a given metric.

MetricsAndChecks class

This class is usually instantiated in the **status** tab of each dataset of folder that needs to be monitored. The right check functions are then called there.

We define the error level of the check during the instantiation of the class, the two possible values being :

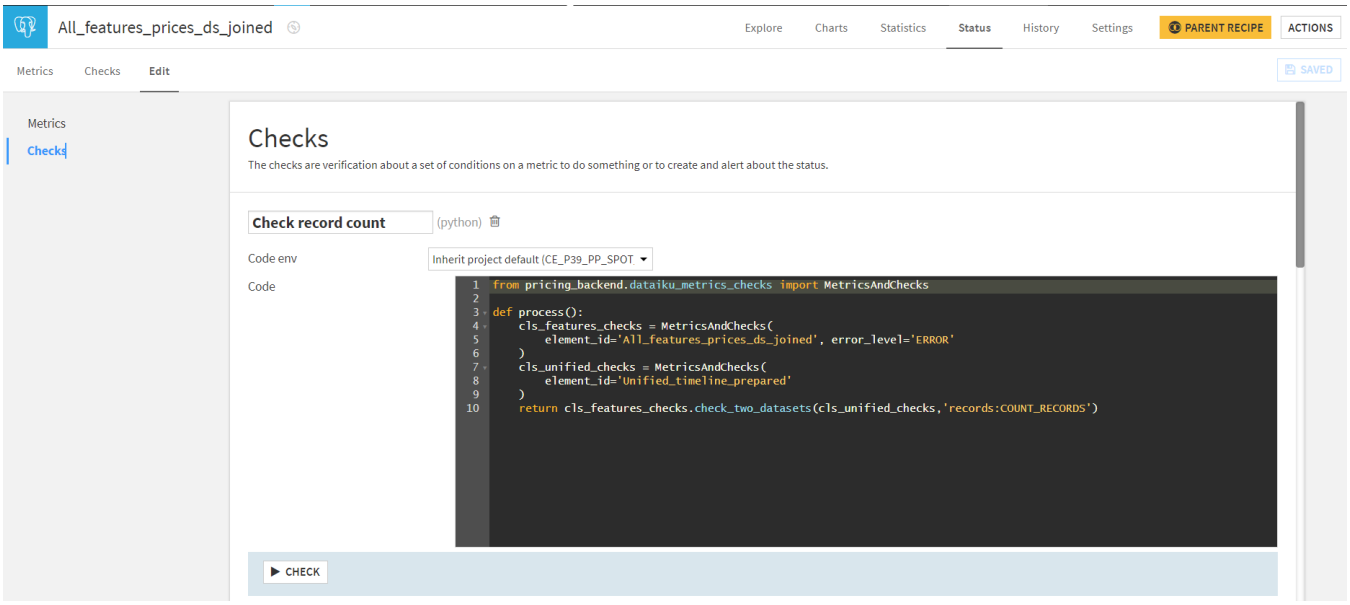
- ERROR when we want a check failure to block the execution of the project
- WARNING otherwise

A global error level can also be defined in the project config by filling the *project_error_level* value of the *metrics_dict* dictionnary. When defined, it will **overr ide all** of the checks error level.

Checks developed in this class are meant to be **generic** in order to be used by several datasets and folders of the project (even applicable to other projects for most of them). This has for consequence that the **returned message will also be quite generic** and will in general display the name of the metric that failed a check with the used threshold. It might be necessary to have in mind the element (dataset or folder) on which it is called to better understand the error.

You can find the list of current checks used in the project here with their related element and error level.

Usage example



From the "All_features_prices_df_joined" dataset, we can access the metrics and checks by using the "Status" tab.

The checks we use are using custom python probe to be able to use our code from the lib.

In the process() function, we can instantiate the MetricsAndChecks class and directly call any of the check developed inside. When calling a check, we also define the error level used and thresholds for the checks that use one.

Note : It is also possible to generate several checks from a single python probe by returning a dictionary if needed.