

Model Metrics

Date published: 2020-07-16

Date modified: 2025-09-30



Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

- Model Metrics..... 4**
 - Enabling model metrics.....4
 - Tracking model metrics without deploying a model.....4
 - Tracking metrics for deployed models..... 5

Model Metrics

Metrics are essential for tracking model performance. By using custom code, you can track specific model predictions and analyze the metrics.

Enabling model metrics

Metrics are used to track the performance of the models. When you enable model metrics while creating a workbench, the metrics are stored in a scalable metrics store. You can track individual model predictions and analyze metrics using custom code.

About this task

Procedure

1. Go to Cloudera AI and click Provision Workbench on the top-right corner.
2. Enter the Cloudera AI Workbench name and other details.
3. Click Advanced Options.
4. Select Enable Model Metrics.

If you want to connect to an external (custom) Postgres database, then specify the details in the additional optional arguments that are displayed. If you do not specify these details, a managed Postgres database will be used to store the metrics.

Tracking model metrics without deploying a model

Cloudera recommends that you develop and test model metrics in a workbench session before actually deploying the model. This workflow avoids the need to rebuild and redeploy a model to test every change.

Metrics tracked in this way are stored in a local, in-memory datastore instead of the metrics database, and are no longer stored when the session exits. You can access these metrics in the same session using the regular metrics API in the `cdsw.py` file.

The following example demonstrates how to track metrics locally within a session, and use the `read_metrics` function to read the metrics in the same session by querying by the time window.

To try this feature in the local development mode, use the following files from the Python template project:

- `use_model_metrics.py`
- `predict_with_metrics.py`

The `predict` function from the `predict_with_metrics.py` file shown in the following example is similar to the function with the same name in the `predict.py` file. It takes input and returns output, and can be deployed as a model. But unlike the function in the `predict.py` file, the `predict` function from the `predict_with_metrics.py` file tracks mathematical metrics. These metrics can include information such as input, output, feature values, convergence metrics, and error estimates. In this simple example, only input and output are tracked. The function is equipped to track metrics by applying the decorator `models.cml_model(metrics=True)`.

```
import pickle
import cml.metrics_v1 as metrics
import cml.models_v1 as models

model = pickle.load(open('model.pkl', 'rb'))
# The model_metrics decorator equips the predict function to
# call track_metrics. It also changes the return type. If the
```

```
# raw predict function returns a value "result", the wrapped
# function will return eg
# {
#   "uuid": "612a0f17-33ad-4c41-8944-df15183ac5bd",
#   "prediction": "result"
# }
# The UUID can be used to query the stored metrics for this
# prediction later.
@models.cml_model(metrics=True)
def predict(args):
    # Track the input.
    metrics.track_metric("input", args)
    # If this model involved features, ie transformations of the
    # raw input, they could be tracked as well.
    # cds.w.track_metric("feature_vars", {"a":1,"b":23})
    petal_length = float(args.get('petal_length'))
    result = model.predict([[petal_length]])

    # Track the output.
    metrics.track_metric("predict_result", result[0][0])
    return result[0][0]
```

You can fetch the metrics from the local, in-memory datastore by using the regular metrics API. To fetch the metrics, set the dev keyword argument to True in the use_model_metrics.py file. You can query the metrics by model, model build, or model deployment using the variables cds.w.dev_model_crn and cds.w.dev_model_build_crn or cds.w.dev_model_deploy_crn respectively.

For example:

```
end_timestamp_ms=int(round(time.time() * 1000))
cds.w.read_metrics(model_deployment_crn=cds.w.dev_model_deployment_crn,
start_timestamp_ms=0,
end_timestamp_ms=end_timestamp_ms,
dev=True)
```

where CRN denotes Cloudera Resource Name, which is a unique identifier from Cloudera, analogous to Amazon's ARN.

Tracking metrics for deployed models

When you have finished developing your metrics tracking code and the code that consumes the metrics, simply deploy the predict function from predict_with_metrics.py as a model. No code changes are necessary.

Calls to read_metrics, track_delayed_metrics, and track_aggregate_metrics need to be changed to take the CRN of the deployed model, build or deployment. These CRNs can be found in the model's **Overview** page.

Calls to the call_model function also requires the model's access key (model_access_key in use_model_metrics.py) from the model's **Settings** page. If authentication has been enabled for the model (the default), a model API key for the user (model_api_token in use_model_metrics.py) is also required. This can be obtained from the user's **Settings** page.