Machine Learning 1.4.1

Model Training and Deployment with Cloudera Machine Learning

Date published: 2020-07-16
Date modified: 2022-07-21

CLOUDERA

Legal Notice

# Contents

Model Training and Deployment Overview

This section provides an overview of model training and deployment using Cloudera Machine Learning.

Machine learning is a discipline that uses computer algorithms to extract useful knowledge from data. There are many different types of machine learning algorithms, and each one works differently. In general however, machine learning algorithms begin with an initial hypothetical model, determine how well this model fits a set of data, and then work on improving the model iteratively. This training process continues until the algorithm can find no additional improvements, or until the user stops the process.

A typical machine learning project will include the following high-level steps that will transform a loose data hypothesis into a model that serves predictions.

1. Explore and experiment with and display findings of data
2. Deploy automated pipelines of analytics workloads
3. Train and evaluate models
4. Deploy models as REST APIs to serve predictions

With Cloudera Machine Learning, you can deploy the complete lifecycle of a machine learning project from research to deployment.

Accelerate Machine Learning Projects from Research to Production ➡️

Explore and Analyze Data	Deploy Automated Pipelines	Train and Evaluate Models	Deploy Models
Workbench (Interactive)	**Jobs** (Batch, non-versioned)	**Experiments** (Batch, versioned)	**Models** (REST API)
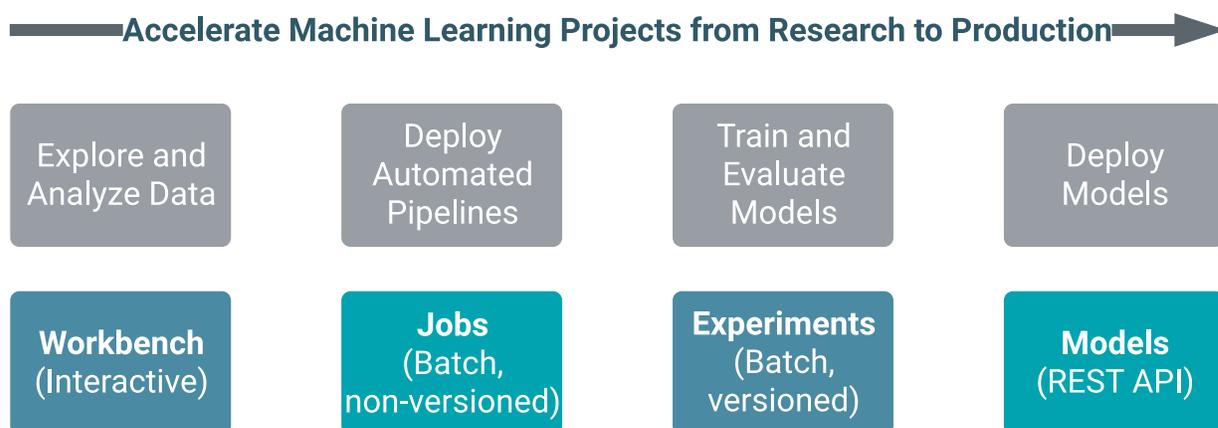
Experiments

This topic introduces you to experiments, and the challenge this feature aims to solve.

Cloudera Machine Learning allows data scientists to run batch experiments that track different versions of code, input parameters, and output (both metrics and files).

Challenge

As data scientists iteratively develop models, they often experiment with datasets, features, libraries, algorithms, and parameters. Even small changes can significantly impact the resulting model. This means data scientists need the ability to iterate and repeat similar experiments in parallel and on demand, as they rely on differences in output and scores to tune parameters until they obtain the best fit for the problem at hand. Such a training workflow requires versioning of the file system, input parameters, and output of each training run.

Without versioned experiments you would need intense process rigor to consistently track training artifacts (data, parameters, code, etc.), and even then it might be impossible to reproduce and explain a given result. This can lead to wasted time and effort during collaboration, not to mention the compliance risks introduced.

Cloudera Machine Learning uses experiments to facilitate ad-hoc batch execution and model training. Experiments are batch executed workloads where the code, input parameters, and output artifacts are versioned. This feature also provides a lightweight ability to track output data, including files, metrics, and metadata for comparison.

Experiments - Concepts and Terminology

This topic walks you through some basic concepts and terminology related to experiments.

The term experiment refers to a non interactive batch execution script that is versioned across input parameters, project files, and output. Batch experiments are associated with a specific project (much like sessions or jobs) and have no notion of scheduling; they run at creation time. To support versioning of the project files and retain run-level artifacts and metadata, each experiment is executed in an isolated container.

Lifecycle of an Experiment
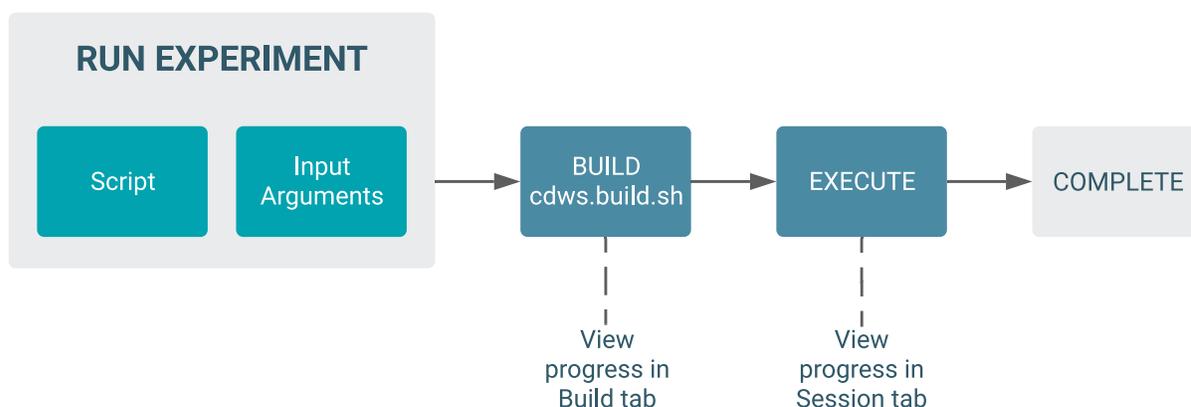
The rest of this section describes the different stages in the lifecycle of an experiment - from launch to completion.

1. **Launch Experiment**

    In this step you will select a script from your project that will be run as part of the experiment, and the resources (memory/GPU) needed to run the experiment. The engine kernel will be selected by default based on your script. For detailed instructions on how to launch an experiment, see *Getting Started with Cloudera Machine Learning*.

2. **Build**

    When you launch the experiment, Cloudera Machine Learning first builds a new versioned engine image where the experiment will be executed in isolation. This new engine includes:

    • the base engine image used by the project (check Project Settings)
    • a snapshot of the project filesystem
    • environmental variables inherited from the project.
    • packages explicitly specified in the project's build script (cdsw-build.sh)

      It is your responsibility to provide the complete list of dependencies required for the experiment via the cdsw-build.sh file. As part of the engine's build process, Cloudera Machine Learning will run the cdsw-build.sh script and install the packages or libraries requested there on the new image.

    For details about the build process and examples on how to specify dependencies, see Engines for Experiments and Models. .

3. **Schedule**

> Once the engine is built the experiment is scheduled for execution like any other job or session. Once the requested CPU/GPU and memory have been allocated to the experiment, it will move on to the execution stage.
>
> Note that if your deployment is running low on memory and CPU, your runs may spend some time in this stage.

4. **Execute**

> This is the stage where the script you have selected will be run in the newly built engine environment. This is the same output you would see if you had executed the script in a session in the Workbench console.
>
> You can watch the execution in progress in the individual run's Session tab.
>
> You can also go to the project Overview Experiments page to see a table of all the experiments launched within that project and their current status.
>
> Run ID: A numeric ID that tracks all experiments launched on a Cloudera Machine Learning deployment. It is not limited to the scope of a single user or project.

Related Information
Running an Experiment with Cloudera Machine Learning

Models

Cloudera Machine Learning allows data scientists to build, deploy, and manage models as REST APIs to serve predictions.

Challenge

Data scientists often develop models using a variety of Python/R open source packages. The challenge lies in actually exposing those models to stakeholders who can test the model. In most organizations, the model deployment process will require assistance from a separate DevOps team who likely have their own policies about deploying new code.

For example, a model that has been developed in Python by data scientists might be rebuilt in another language by the devops team before it is actually deployed. This process can be slow and error-prone. It can take months to deploy new models, if at all. This also introduces compliance risks when you take into account the fact that the new re-developed model might not be even be an accurate reproduction of the original model.

Once a model has been deployed, you then need to ensure that the devops team has a way to rollback the model to a previous version if needed. This means the data science team also needs a reliable way to retain history of the models they build and ensure that they can rebuild a specific version if needed. At any time, data scientists (or any other stakeholders) must have a way to accurately identify which version of a model is/was deployed.

Solution

Cloudera Machine Learning allows data scientists to build and deploy their own models as REST APIs. Data scientists can now select a Python or R function within a project file, and Cloudera Machine Learning will:

- Create a snapshot of model code, model parameters, and dependencies.
- Package a trained model into an immutable artifact and provide basic serving code.
- Add a REST endpoint that automatically accepts input parameters matching the function, and that returns a data structure that matches the function's return type.
- Save the model along with some metadata.
- Deploy a specified number of model API replicas, automatically load balanced.

Models - Concepts and Terminology

Model

Model is a high level abstract term that is used to describe several possible incarnations of objects created during the model deployment process. For the purpose of this discussion you should note that 'model' does not always refer to a specific artifact. More precise terms (as defined later in this section) should be used whenever possible.

Stages of the Model Deployment Process
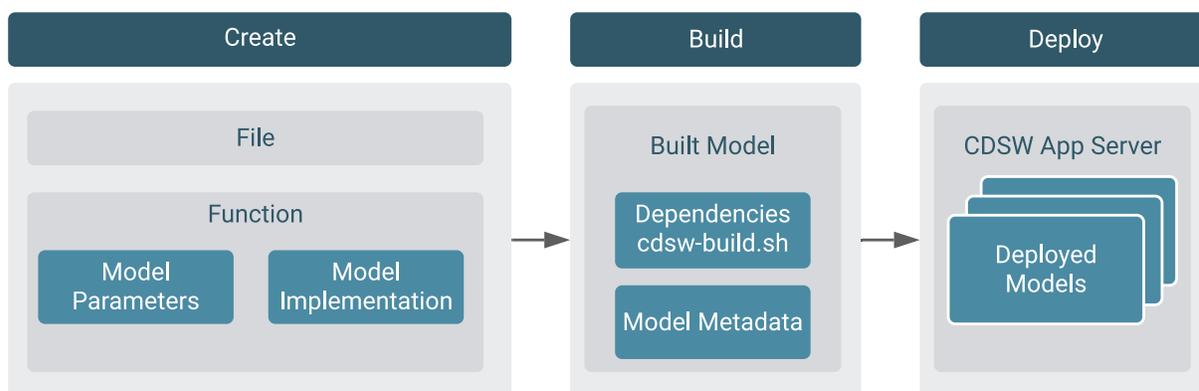
The rest of this section contains supplemental information that describes the model deployment process in detail.

Create

- File - The R or Python file containing the function to be invoked when the model is started.
- Function - The function to be invoked inside the file. This function should take a single JSON-encoded object (for example, a python dictionary) as input and return a JSON-encodable object as output to ensure compatibility with any application accessing the model using the API. JSON decoding and encoding for model input/output is built into Cloudera Machine Learning.

  The function will likely include the following components:

  - Model Implementation

    The code for implementing the model (e.g. decision trees, k-means). This might originate with the data scientist or might be provided by the engineering team. This code implements the model's predict function, along with any setup and teardown that may be required.
  - Model Parameters

    A set of parameters obtained as a result of model training/fitting (using experiments). For example, a specific decision tree or the specific centroids of a k-means clustering, to be used to make a prediction.

Build

This stage takes as input the file that calls the function and returns an artifact that implements a single concrete model, referred to as a model build.

- Built Model

  A built model is a static, immutable artifact that includes the model implementation, its parameters, any runtime dependencies, and its metadata. If any of these components need to be changed, for example, code changes to the implementation or its parameters need to be

retrained, a new build must be created for the model. Model builds are versioned using build numbers.

To create the model build, Cloudera Machine Learning creates a Docker image based on the engine designated as the project's default engine. This image provides an isolated environment where the model implementation code will run.

To configure the image environment, you can specify a list of dependencies to be installed in a build script called cdsw-build.sh.

For details about the build process and examples on how to install dependencies, see *Engines for Experiments and Models*.

- Build Number:

Build numbers are used to track different versions of builds within the scope of a single model. They start at 1 and are incremented with each new build created for the model.

Deploy

This stage takes as input the memory/CPU resources required to power the model, the number of replicas needed, and deploys the model build created in the previous stage to a REST API.

- Deployed Model

A deployed model is a model build in execution. A built model is deployed in a model serving environment, likely with multiple replicas.

- Environmental Variable

You can set environmental variables each time you deploy a model. Note that models also inherit any environment variables set at the project and global level. (For more information see *Engine Environment Variables*.) However, in case of any conflicts, variables set per-model will take precedence.

> **Note:** If you are using any model-specific environmental variables, these must be specified every time you re-deploy a model. Models do not inherit environmental variables from previous deployments.

- Model Replicas

The engines that serve incoming requests to the model. Note that each replica can only process one request at a time. Multiple replicas are essential for load-balancing, fault tolerance, and serving concurrent requests. Cloudera Machine Learning allows you to deploy a maximum of 9 replicas per model.

- Deployment ID

Deployment IDs are numeric IDs used to track models deployed across Cloudera Machine Learning. They are not bound to a model or project.

Related Information

Experiments - Concepts and Terminology
Engines for Experiments and Models
Engines Environment Variables