BigML Documentation

Release 1.0.2

The BigML Team

Nov 29, 2022

CONTENTS

All the resources generated by the *BigML* API-first platform, including models, are totally white-box, and they can be downloaded as JSON and used to predict anywhere.

On the other hand, MLFlow offers tracking and deploying capacities for a variety of ML models as long a *flavor* is created to define how to log, save and load those models to be actionable.

The *bigmlflow* library implements this flavor. It uses BigML's Python bindings to integrate the BigML models with *MLFlow's* tracking and deploying capacities.

CHAPTER

ONE

INSTALLATION

This library is available as a PyPI package. To install it, just run:

```
pip install bigmlflow
```

The flavor is implemented in a single `bigmlflow` module

CHAPTER

TWO

FLAVOR METHODS

The bigmlflow.bigml module provides an API for logging and loading BigML models. This module exports BigML models with the following flavors:

BigML (native) format This is the main flavor that can be loaded back into BigML.

mlflow.pyfunc Produced for use by generic pyfunc-based deployment tools and batch inference.

bigmlflow.bigml.**get_default_conda_env**()

Returns The default Conda environment for MLflow Models produced by calls to *save_model()* and *log_model()*.

bigmlflow.bigml.**get_default_pip_requirements**()

Returns A list of default pip requirements for MLflow Models produced by this flavor. Calls to *save_model()* and *log_model()* produce a pip environment that, at minimum, contains these requirements.

bigmlflow.bigml.**load_model**(*model_uri*, *dst_path=None*)

Load a BigML model from a local file (if run_id is None) or a run.

Parameters

- **model_uri** – The location, in URI format, of the MLflow model. For example:
  - /Users/me/path/to/local/model
  - relative/path/to/local/model
  - s3://my_bucket/path/to/model
  - runs:/<mlflow_run_id>/run-relative/path/to/model
  - models:/<model_name>/<model_version>
  - models:/<model_name>/<stage>

  For more information about supported URI schemes, see Referencing Artifacts.
- **dst_path** – The local filesystem path to which to download the model artifact. This directory must already exist. If unspecified, a local output path will be created.

Returns A SupervisedModel model object.

bigmlflow.bigml.**log_model**(*bigml_model*, *artifact_path*, *conda_env=None*, *code_paths=None*, *registered_model_name=None*, *signature: mlflow.models.signature.ModelSignature = None*, *input_example: Union[pandas.core.frame.DataFrame, numpy.ndarray, dict, list, csr_matrix, csc_matrix] = None*, *pip_requirements=None*, *extra_pip_requirements=None*, ***kwargs*)

Log a BigML model as an MLflow artifact for the current run.

Parameters

- **bigml_model** – BigML model to be saved.
- **artifact_path** – Run-relative artifact path.
- **conda_env** – Either a dictionary representation of a Conda environment or the path to a conda environment yaml file. If provided, this describes the environment this model should be run in. At minimum, it should specify the dependencies contained in *get_default_conda_env()*. If `None`, a conda environment with pip requirements inferred by `mlflow.models.infer_pip_requirements()` is added to the model. If the requirement inference fails, it falls back to using *get_default_pip_requirements()*. pip requirements from `conda_env` are written to a pip `requirements.txt` file and the full conda environment is written to `conda.yaml`. The following is an *example* dictionary representation of a conda environment:

```
{
    "name": "mlflow-env",
    "channels": ["conda-forge"],
    "dependencies": [
        "python=3.7.0",
        {
            "pip": [
                "bigml==x.y.z"
            ],
        },
    ],
}
```

- **code_paths** – A list of local filesystem paths to Python file dependencies (or directories containing file dependencies). These files are *prepended* to the system path when the model is loaded.
- **registered_model_name** – If given, create a model version under `registered_model_name`, also creating a registered model if one with the given name does not exist.
- **signature** – `ModelSignature` describes model input and output `Schema`. The model signature can be `inferred` from datasets with valid model input (e.g. the training dataset with target column omitted) and valid model output (e.g. model predictions generated on the training dataset), for example:

```
from mlflow.models.signature import infer_signature
train = df.drop_column("target_label")
predictions = ... # compute model predictions
signature = infer_signature(train, predictions)
```

- **input_example** – Input example provides one or several instances of valid model input. The example can be used as a hint of what data to feed the model. The given example will be

converted to a Pandas DataFrame and then serialized to json using the Pandas split-oriented format. Bytes are base64-encoded.

- **pip_requirements** – Either an iterable of pip requirement strings (e.g. `["bigml", "-r requirements.txt", "-c constraints.txt"]`) or the string path to a pip requirements file on the local filesystem (e.g. `"requirements.txt"`). If provided, this describes the environment this model should be run in. If `None`, a default list of requirements is inferred by `mlflow.models.infer_pip_requirements()` from the current software environment. If the requirement inference fails, it falls back to using *get_default_pip_requirements()*. Both requirements and constraints are automatically parsed and written to `requirements.txt` and `constraints.txt` files, respectively, and stored as part of the model. Requirements are also written to the `pip` section of the model's conda environment (`conda.yaml`) file.

- **extra_pip_requirements** – Either an iterable of pip requirement strings (e.g. `["pandas", "-r requirements.txt", "-c constraints.txt"]`) or the string path to a pip requirements file on the local filesystem (e.g. `"requirements.txt"`). If provided, this describes additional pip requirements that are appended to a default set of pip requirements generated automatically based on the user's current software environment. Both requirements and constraints are automatically parsed and written to `requirements.txt` and `constraints.txt` files, respectively, and stored as part of the model. Requirements are also written to the `pip` section of the model's conda environment (`conda.yaml`) file.

  Warning: The following arguments can't be specified at the same time:

  - `conda_env`
  - `pip_requirements`
  - `extra_pip_requirements`

  This example demonstrates how to specify pip requirements using `pip_requirements` and `extra_pip_requirements`.

- **kwargs** – kwargs to pass to `bigml_model` save model method, if any.

Returns A `ModelInfo` instance that contains the metadata of the logged model.

bigmlflow.bigml.**save_model**(*bigml_model*, *path*, *conda_env=None*, *code_paths=None*, *mlflow_model=None*, *signature: mlflow.models.signature.ModelSignature = None*, *input_example: Union[pandas.core.frame.DataFrame, numpy.ndarray, dict, list, csr_matrix, csc_matrix] = None*, *pip_requirements=None*, *extra_pip_requirements=None*)

Save an BigML model to a path on the local file system.

Parameters

- **bigml_model** – BigML model to be saved.
- **path** – Local path where the model is to be saved.
- **conda_env** – Either a dictionary representation of a Conda environment or the path to a conda environment yaml file. If provided, this describes the environment this model should be run in. At minimum, it should specify the dependencies contained in *get_default_conda_env()*. If `None`, a conda environment with pip requirements inferred by `mlflow.models.infer_pip_requirements()` is added to the model. If the requirement inference fails, it falls back to using *get_default_pip_requirements()*. pip requirements from `conda_env` are written to a pip `requirements.txt` file and the full conda environment is written to `conda.yaml`. The following is an *example* dictionary representation of a conda environment:

```
{
    "name": "mlflow-env",
    "channels": ["conda-forge"],
    "dependencies": [
        "python=3.7.0",
        {
            "pip": [
                "bigml==x.y.z"
            ],
        },
    ],
}
```

- **code_paths** – A list of local filesystem paths to Python file dependencies (or directories containing file dependencies). These files are *prepended* to the system path when the model is loaded.
- **signature** – `ModelSignature` describes model input and output `Schema`. The model signature can be `inferred` from datasets with valid model input (e.g. the training dataset with target column omitted) and valid model output (e.g. model predictions generated on the training dataset), for example:

```
from mlflow.models.signature import infer_signature
train = df.drop_column("target_label")
predictions = ... # compute model predictions
signature = infer_signature(train, predictions)
```

- **input_example** – Input example provides one or several instances of valid model input. The example can be used as a hint of what data to feed the model. The given example will be converted to a Pandas DataFrame and then serialized to json using the Pandas split-oriented format. Bytes are base64-encoded.
- **mlflow_model** – `mlflow.models.Model` this flavor is being added to.

CHAPTER

THREE

BIGMLFLOW USAGE

Some examples are available in the repository to illustrate how you can use *MLFlow* to generate *BigML* models, log evaluation metrics and deploy the different Supervised Models available in the *BigML* platform.

CHAPTER

FOUR

TESTS

The *tests* directory contains some tests for the logging, saving and loading of models. We use *Pytest* to run the tests, so you can install it separately

```
pip install pytest
```

or as an extra for development and testing purposes

```
pip install -e .[tests]
```

CHAPTER

FIVE

HOW TO CONTRIBUTE

Please follow the next steps:

1. Fork the project on github.com.
2. Create a new branch.
3. Commit changes to the new branch.
4. Send a pull request.

PYTHON MODULE INDEX

b

INDEX